A SURVEY OF PUBLIC-KEY INFRASTRUCTURES

by

Marc Branchaud

March 1997
Copy 0

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements of the degree of

Master of Science in Computer Science

Department of Computer Science
McGill University, Montreal

ABSTRACT

Public-key cryptography is fast becoming the foundation for online commerce and other applications that require security and authentication in an open network. The widespread use of public-key cryptography requires a public-key infrastructure to publish and manage public-key values. Without a functioning infrastructure, public-key cryptography is only marginally more useful than traditional, secret-key cryptography.

This thesis presents a set of characteristics that are common to all public-key infrastructures. These criteria are intended to encapsulate the fundamental issues that arise when dealing with such systems. They can be used both as a “shopping list” for those who need to choose an infrastructure for a particular application, and as a guide for infrastructure developers, that they may be more aware of any compromises or tradeoffs they might make in their work.

The characteristics are used to present a survey of current and some proposed infrastructure systems. The criteria reveal the strengths and weaknesses of each system, and indicate where improvements may be required.

The characteristics presented here are intended to enhance rather than restrict development in the field. This is not necessarily an exhaustive list, and it is the author’s intention to revise these criteria as new ideas emerge.

RESUME

La cryptographie à clé publique s'impose rapidement comme l'élément de base du commerce virtuel et d'autres applications exigeant des protocoles de sécurité et d'authentification dans un réseau ouvert. Son utilisation par le plus grand nombre nécessite une infrastructure permettant la publication et la gestion de clés publiques. Sans une infrastructure efficace, la cryptographie à clé publique ne saurait véritablement rendre de plus grands services que les méthodes classiques de cryptographie à clé secrète.

La présente thèse propose un ensemble de caractéristiques communes à toutes les infrastructures de clés publiques, afin de résumer les problèmes fondamentaux que peuvent poser des systèmes de cette nature. Les personnes devant choisir une infrastructure convenant à une application en particulier pourront donc s'y reporter, tandis que les créateurs d'infrastructures y trouveront un aperçu des compromis qu'ils pourraient être tenus d'accepter.

L'énoncé de ces caractéristiques correspond également à un survol des infrastructures existantes ainsi que de certains modèles à l'étude. Ces critères font ressortir les points forts et les points faibles de chaque système ainsi que les améliorations souhaitables.

Ces caractéristiques sont présentées dans le but d'améliorer les progrès dans ce domaine, et non de les restreindre. La liste n'est pas forcément exhaustive et l'auteur exprime l'intention de revoir ses critères et d'y intégrer les plus récents développements dans le domaine.

TABLE OF CONTENTS

INTRODUCTION **1**

ORIGINALITY OF WORK 2

PKI CRYPTOGRAPHY BASICS **4**

SECRET-KEY CRYPTOGRAPHY 4
PUBLIC-KEY CRYPTOGRAPHY 4
DIGITAL SIGNATURES 6
HASH FUNCTIONS 7
SUMMARY 8

BASIC PUBLIC-KEY INFRASTRUCTURE CHARACTERISTICS **9**

WHAT IS A PUBLIC-KEY INFRASTRUCTURE? 9
CERTIFICATION 9
CA ARRANGEMENTS 11
ENTITY RELATIONSHIPS 15
VALIDATION 16
AUTHENTICATION 18
Limitations of PKI Authentication *19*
ANONYMITY 20
SUMMARY 21

A WORLD WITHOUT PUBLIC-KEY INFRASTRUCTURES **23**

INSECURE EMAIL 23
FTP AND ACCESS CONTROL 25

PRETTY GOOD PRIVACY **27**

INTRODUCTION 27
PGP IN ACTION 28
THE PRETTY GOOD PRIVACY PKI 29

X.509 .... **32**

INTRODUCTION 32
THE X.509 STANDARD 33
X.500 .... *33*
X.509 Versions 1 and 2 .... *34*
X.509 Version 3 .... *37*
Version 3 Certificate Extensions 37
Version 3 CRL Extensions 39
Object Registration *40*
BASIC PKI CHARACTERISTICS OF X.509 42
X.509 ON THE INTERNET 43
Privacy Enhanced Mail *44*
PEM in Action 46
The Privacy Enhanced Mail PKI 47
CONCLUSIONS 48

INTERNET DOMAIN NAME SYSTEM SECURITY EXTENSIONS **50**

INTRODUCTION 50

Overview of the DNS 50
DNS Security Extensions 52
Secure DNS in Action 53
The Secure DNS PKI 55
Conclusions 57

CREDENTIAL-BASED PKI SYSTEMS 59

Simple Distributed Security Infrastructure (SDSI) 59
SDSI Certificates 59
SDSI Names 60
SDSI Groups 61
SDSI in Action 62
Key Management Under SDSI 64
The SDSI PKI 65
The Simple Public Key Infrastructure 67
Conclusions 67

CONCLUSION 69

LIST OF FIGURES

Number *Page*

Figure 1 – Message encryption using a secret key (S) to encode the message and a public key (P) to encode the secret key....5
Figure 2 – Creating a digital signature....7
Figure 3 – Basic public-key cryptography message formats....8
Figure 4 – A basic certificate....10
Figure 5 – A certification path from Alice to Bob....12
Figure 6 – A general hierarchy with cross-certificates....13
Figure 7 – Top-down hierarchy....14
Figure 8 – Internet communications....24
Figure 9 – The FTP protocol....25
Figure 10 – Pretty Good Privacy....28
Figure 11 – The X.500 Directory Information Tree....33
Figure 12 – The X.509 version 2 certificate....34
Figure 13 – X.509v1 CRL format....36
Figure 14 – A progressive-constrained trust chain....38
Figure 15 – Object registration example....40
Figure 16 – The PEM CA hierarchy....44
Figure 17 – PEM example....46
Figure 18 – The DNS hierarchy....50
Figure 19 – Secure DNS example....54
Figure 20 – SDSI protocol example....62
Figure 21 – Sample SDSI groups....63

LIST OF TABLES

Number *Page*

Table 1 – Basic PKI Characteristics....22
Table 2 – Basic Characteristics of the PGP PKI....31
Table 3 – Basic Characteristics of the X.509 PKI....43
Table 4 – Basic Characteristics of the PEM PKI....48
Table 5 – Basic Characteristics of the DNS PKI....57
Table 6 – Basic Characteristics of the SDSI PKI....66

ACKNOWLEDGMENTS

The author wishes to thank Warwick Ford for his generous assistance in giving me access to X.509 documentation as well as his excellent explanations of the X.509 standard, and especially his patient replies to my queries. I am also grateful to many others who have contributed indirectly, by making their work freely available on the Internet. This thesis would not have been possible without their willingness to openly share their work. I also wish to thank my supervisor, David Avis, for allowing me the chance to do this work and for his sage guidance and astute advice.

To my wife Sony, my one and only.

Chapter 1

INTRODUCTION

Computers play an increasingly larger role in everyday life. From the embedded microprocessors found in virtually every electronic appliance, to the escalating number of personal computers used for business, entertainment and education, Nicholas Negroponte's statement that "computing is not about computers … it is about living"[1] is becoming truer by the day. Now, with the recent explosive growth of the Internet, all these computers are becoming interconnected in a global communications network. Many view the Internet as a universal communications medium that can replace telephone, television and radio. The potential is there, but progress has been hampered by the open design of the network itself. It is still too easy to intercept, monitor and forge messages on the Internet, and people are reluctant to use the network for financially or legally sensitive data.

The problems faced by users of the Internet fall into two main categories: *privacy* and *authentication*. Privacy involves transmitting messages that cannot be altered or read en route, while authentication allows each party to a communication to be sure of the identity of the other (i.e. messages can't be forged). Cryptography holds the promise of a solution to these problems. Cryptography is the science of secret writing. It provides a means whereby two people (or their computers), commonly designated Alice and Bob, can communicate openly in such a way that a third party, usually named Oscar, is unable to determine or alter what is being said. By assuring privacy, cryptography indirectly provides authentication because only Alice and Bob know how to encrypt and decipher each other's messages.

---

1 [Ne95], page 6. Negroponte goes on to predict that "Early in the next millennium your right and left cuff links or earrings may ... have more computer power than your present PC."

A form of cryptography known as *public-key cryptography* appears to be best suited to fulfilling the requirements of the Internet. Each user of a public-key cryptosystem holds a pair of related keys. Anything encoded with one key can only be decoded by it's counterpart. Each user keeps one key secret and publishes the other. Thus other people can employ the user's public key to send messages that only the user can read, or the user can "sign" a message with her private key to authenticate it – other people can apply the user's public key to verify that the message came from the user.

Crucial to the operation of a global public-key cryptosystem on the Internet is a practical and reliable means of publishing the public keys, called a Public-Key Infrastructure or PKI. There are as yet only a handful proposals for an Internet PKI,[2] many of which are still in draft form, and no single one has yet to gain widespread use on the network. Indeed, many feel that, for the near future, there will be several PKI systems operating and inter-operating on the Internet.

This thesis presents a set of basic PKI characteristics that apply to any PKI system, and uses these characteristics to describe Internet PKI proposals. It is hoped that these characteristics will prove useful both as a guide to PKI designers and as an aid to PKI implementers in deciding which PKI system best suits their needs.

We begin in Chapter 2 with a short discussion of the basic elements of a PKI: private- and public-key cryptosystems, digital signature systems and message hashing algorithms. In Chapter 3 we describe PKIs in general, their requirements and limitations, and we present the basic PKI characteristics. Chapter 4 discusses the current operation of the Internet without a PKI through two examples: email and FTP. We next turn to current and proposed Internet PKI systems. Chapter 5 deals with Pretty Good Privacy. Chapter 6 covers X.509-based proposals. Chapter 7 is devoted to the Secure DNS PKI. Chapter 8 discusses recent ideas for credential- or attribute-based systems.

[2] Specifically, these are PGP, PEM, PKIX, Secure DNS, SPKI and SDSI.

Originality of Work

Here is a breakdown of the sources for the material in this thesis. All work is original except where indicated here and in the actual text. Any application of the basic PKI characteristics to the various PKIs discussed is original. All of the *“**… in Action**”* subsections are original.

- The discussion of basic cryptography in Chapter 2 is distilled from [St95], although all the figures are original. The discussion of the importance of having separate signature and encryption keys is adopted from [FoBa].
- Most of Chapter 3, especially the ten basic characteristics, is original. Most of the *italicized* terms defined are common to the field, although the phrase “CRL time-granularity problem” is original. Many of the definitions, as well as figures 5, 6 and 7, are adapted from [FoBa].
- Chapter 4 is original.
- The description of PGP in Chapter 5 is derived from [Zimm]. The critique of the PGP PKI is original.
- Chapter 6’s discussion of the X.509 and PEM standards is derived from [FoBa] and [RFC1422], including all the figures. The discussion of the implications of object identifiers, and the description of figure 11, is original.
- Chapter 7’s description of the Internet domain name system and its security extensions is derived from the appropriate Internet RFC documents.
- The description of SDSI in chapter 8 is derived from [SDSI]. The sections following and including ***SDSI in Action*** are original.
- Chapter 9 is original.

Chapter 2

PKI CRYPTOGRAPHY BASICS

This chapter provides a cursory overview of the cryptographic techniques that make up a PKI. We focus here on the general properties of these techniques, as an in-depth discussion of each method's various schemes is beyond the scope of this thesis. For more rigorous discussion, refer to a recent book on cryptography, such as [St95] or [Sc96].

Secret-Key Cryptography

Secret-key cryptography[3] is the classical form of cryptography that has been around since ancient times. With a secret-key cryptosystem, Alice and Bob share a secret: the key used for encryption and decryption. This requires prior communication between Alice and Bob over a secure channel, so that they may agree on a key. There are a great many secret-key systems, the best-known probably being the Data Encryption Standard (DES, and it's newer counterpart Triple-DES) [DES].

There exist systems for communicating securely over public networks using only secret-key cryptography, most notably MIT's Kerberos system ([RFC1510]). However, these schemes do not scale well to large, inter-organizational populations, and they also carry extra security procedures that public-key systems do not need, such as storing the secret keys on a secure, central server. Still, as we shall see below, secret-key systems have their place in a PKI.

[3] Also commonly called *private-key* cryptography. We prefer the term "secret-key" as it avoids confusion with the private keys used in public-key systems.

Public-Key Cryptography

In contrast with secret-key cryptography, public-key cryptography is very new. It was first conceived in 1976 by Diffie and Hellman ([DH76]), and in 1977 Rivest, Shamir and Adleman invented the RSA Cryptosystem ([RSA78]), the first realization of a public-key system. There have since been several proposals for public-key schemes, including the ElGamal Cryptosystem ([El85]) and elliptic curve cryptosystems ([Sa96]).

Each public-key cryptosystem has its own technical nuances, however they all share the same basic property that given an encryption key it is computationally infeasible to determine the decryption key (and vice-versa). This property lets a user, Alice, publish her encryption key. Anyone can use that public key to encrypt a message that only Alice can decipher with her private key. We say that Alice "owns" the "key-pair."

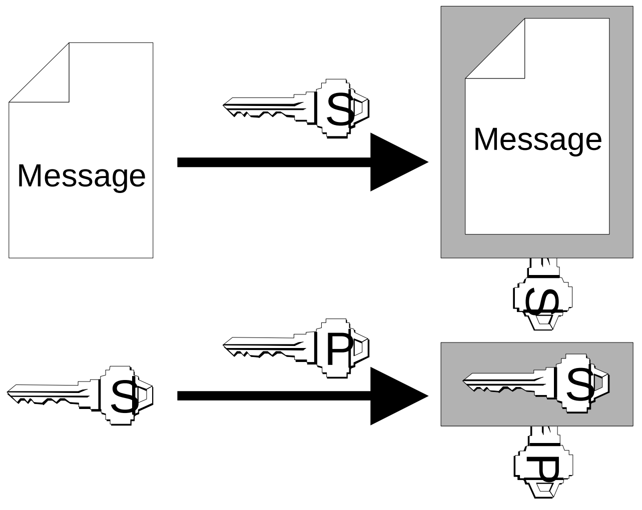

Figure 1 – Message encryption using a secret key (S) to encode the message and a public key (P) to encode the secret key

In practice, computing a public-key cipher takes much longer than encoding the same message with a secret-key system.[4] This has lead to the practice of encrypting messages with a secret-key system such as DES, then encoding the secret key itself with a public-key system such as RSA (see Figure 11). We say that the public-key system "transports" the secret key. Since the secret key is usually much shorter than the message, this technique results in significantly faster processing than if public-key cryptography alone were used.

[4] In [St95] page 128: "RSA is roughly 1500 times slower than DES."

Thus each securely-transmitted message has two components: the message proper (encoded with a secret-key system) and the key used to encode the message (itself encoded using a public-key system). Reading the message is hence a two step process: first decode the secret key, then decode the message. In this thesis, when we say that a person used a public (or private) key to encrypt a message, or that a message is encrypted, we are referring to this combined technique.

Digital Signatures

The very nature of public-key cryptography permits a form of message signing. Suppose Alice publishes her decryption key and keeps her encryption key secret. When Alice encrypts a message, anyone can decrypt it using her public decrypting key and, in doing so, they can be sure that the message could only have been encrypted by Alice, since she is the sole possessor of her encryption key. Alice has effectively “signed” the message.

Some public-key cryptosystems, such as RSA, have the property that both the public and private keys can be used for encryption and decryption. In other words, one key pair can be used for both message encryption and digital signature. This practice, however, creates a number of problems with respect to the management of the key pair. For example, consider the archival requirements of the private key under each circumstance.

For a key pair used for digital signatures, the private key should never be backed up, and it should be destroyed at the end of its active life. If the private key is ever disclosed it can be used to forge documents. Even if its value is discovered long after its active life has ended, it can still be used to forge signatures on ostensibly-old documents.

In contrast, with a key pair used for encryption the private key should be archived for as long as possible, because if the private key is ever lost it would be impossible to retrieve messages encrypted with its public counterpart. It is therefore sensible to keep multiple copies of this private key. Since this contradicts the archiving requirements of a signature private key, one is better off in keeping separate key pairs for each function.

[FoBa] discusses these issues in greater depth. For our purposes, we will always assume that the encrypting key pair is distinct from the signature key pair.

Hash Functions

Typically, to digitally sign a message, rather than encrypt the message using a public-key scheme, the message is *hashed* using a cryptographic hash function, and the hash is encrypted (see Figure 22). A cryptographic hash function maps an arbitrary-length message to a fixed number of bits. Hash functions have the following properties:

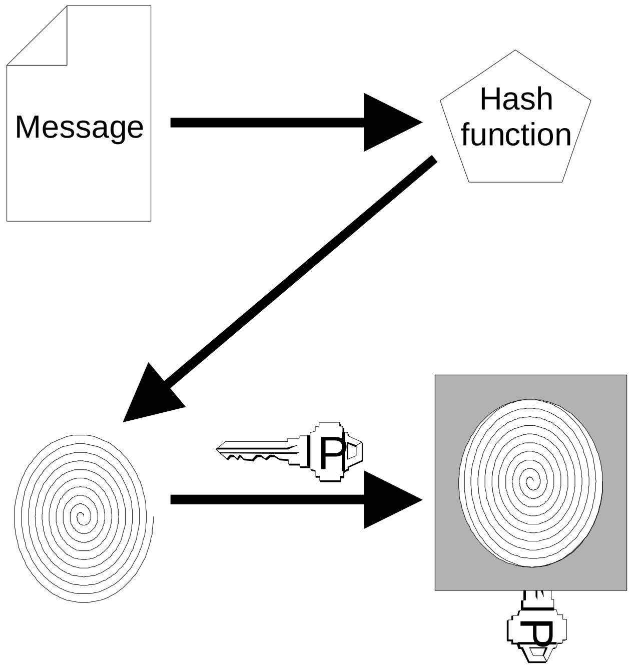

Figure 2 – Creating a digital signature

- They are *collision-free*: it is computationally infeasible to find two different messages that have the same hash.
- They are *one-way*: given a message hash, it is computationally infeasible to find *any* message with the same hash value.

The first property in fact implies the second;[5] we list both to better illustrate the concept. Hash functions are also called *message digest* or *fingerprint* algorithms. Some better-known examples are MD5 ([RFC1321]) and SHA-1 ([SHS]).

As we stated above, digitally signing a message using hashes is a two-step process. The message is first hashed, then the hash result is encrypted using a public-key scheme. Then the message is transmitted along with its encrypted hash. To verify the signature, the recipient needs to hash the message himself, then decrypt the transmitted hash and compare the pair of hash values. The signature is valid if the two values match, otherwise the message was somehow altered, perhaps maliciously, in transit.

[5] See [St95] page 235.

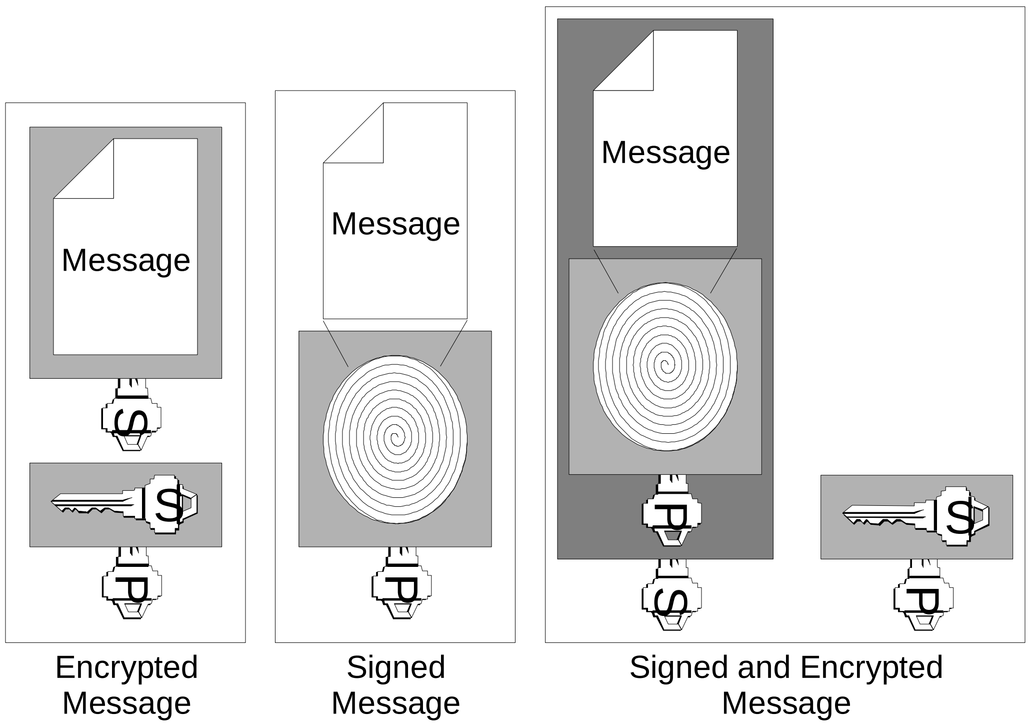

Figure 3 – Basic public-key cryptography message formats

Summary

Figure 33 summarizes the basic formats of messages when public-key cryptography is used:

- An *encrypted* message, in which a symmetric key encrypts the message and a public key encrypts the symmetric key.
- A *signed* message, in which the message is hashed and the hash is encrypted with a public key.
- A *signed and encrypted* message, in which the message is signed using the private key of the sender, then the signed message is encrypted using the public key of the recipient.

Chapter 3

BASIC PUBLIC-KEY INFRASTRUCTURE CHARACTERISTICS

In this chapter we provide a working definition of "public-key infrastructure" and discuss the characteristics of PKIs in general. We propose ten basic characteristics common to all PKIs. The concepts described here provide the basis for understanding and evaluating public-key infrastructure systems, which are discussed in subsequent chapters.

What is a Public-Key Infrastructure?

In its most simple form, a PKI is a system for publishing the public-key values used in public-key cryptography. There are two basic operations common to all PKIs:

- *Certification* is the process of binding a public-key value to an individual, organization or other entity, or even to some other piece of information, such as a permission or credential.
- *Validation* is the process of verifying that a certification is still valid.

How these two operations are implemented is the basic defining characteristic of all PKIs. We now describe in general terms the various methods employed to perform these operations, and discuss the various issues that result from their use. As we proceed, we will point out the basic characteristics of PKIs. These are summarized in Table 11 at the end of this chapter.

Certification

Certification is the fundamental function of all PKIs. It is the means by which public-key values, and information pertaining to those values, are published. For our purposes, we define a *certificate* as the form in which a PKI communicates public key values or information about public keys, or both.

This is a very broad definition of a certificate. At its most basic, a certificate is merely a public key value. In more traditional terms, a certificate is a collection of information that has been digitally signed by its issuer (see Figure 44). Such certificates are distinguished by the kind of information they contain.

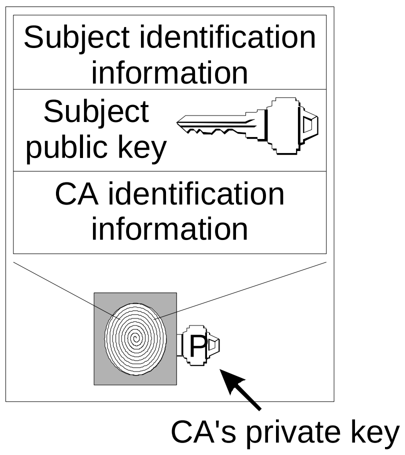
Subject identification information
Subject public key
CA identification information
P
CA's private key

Figure 4 – A basic certificate

An *identity certificate* simply identifies an entity, called the certificate *subject*, and lists the public-key value(s) for that entity.[6] A *credential certificate* describes non-entities, such as a permission or credential. This is discussed further below under **Authentication**.

A certificate *user* is an entity who relies upon the information contained in a certificate. The certificate user trusts the issuing authority to issue "true" certificates. That is, certificates which truly identify the subject and its public key (in the case of identity certificates), or which truly describe a subject's credentials (in the case of credential certificates). The certificate issuer is commonly called a *certification authority* (CA).

To help illustrate these concepts, we present an example using identity certificates. Imagine that Alice wishes to securely communicate with Bob using a public key cryptosystem. Alice needs to know the value of Bob's public encrypting key. Without a PKI, Alice must have direct knowledge of that key, i.e. Bob must com-

[6] An entity in this context can be an individual, corporation, government or other organization. It is easiest to think of an entity as some person or party who can control a private key.

municate it to her via a secure channel. If Alice also wishes to communicate with Doug, she must also have direct knowledge of Doug's public encrypting key.

With a PKI, Alice only needs to have direct knowledge of a CA's public signing key. The CA would issue an identity certificate for each of Bob's and Doug's public encrypting keys. Then if Alice wishes to communicate with Bob or Doug, she can use the appropriate certificate to obtain the correct public key value. In this case, Alice is the certificate user while Bob and Doug are both the subjects of different certificates.

The information contained in a certificate is a basic characteristic of different PKIs. As well, the relationship between the CA, the certificate user and the certificate subject forms another basic PKI characteristic. All three may be distinct entities, such as in the above example, or any two (or all three) can be the same entity. The trust relationships between the three also form a third basic PKI characteristic. In the above example, Alice is required to trust the CA's certificates. If Alice and the CA are distinct entities, how Alice trusts the CA will define how much confidence she has in using the CA's certificates for secure communications.

CA Arrangements

It is obviously impractical to have a single CA act as the authority for the entire world. Therefore, most PKIs permit CAs to certify other CAs. In effect one CA is telling its users that they can trust what a second CA says in its certificates. Returning to our example above, Alice, Bob and Doug would typically each be certified by a different CA. For Alice to then communicate with Bob, she would either need direct knowledge of Bob's CA's signature public key, or Alice's CA could issue a certificate for that key. Then Alice could securely obtain Bob's public key while only having direct knowledge of her CA's key. In this case, the certificates issued for Alice and Bob are called *end-user certificates* while the certificate issued by Alice's CA for Bob's CA is called a *CA-certificate*.

In general, there may be an arbitrary number of CAs on a path between Alice and Bob (see Figure 55). To obtain Bob’s public key, Alice would have to verify the certificate of each CA in turn until she obtained Bob’s certificate. This process is called *certification path validation*. The length of the certification path is the number of CAs between Alice and Bob, or the number of certificates Alice needs to verify in obtaining Bob’s key. The path in Figure 55 is made up of three certificates: two CA-certificates and one end-user certificate. Certificate 1 is a CA-certificate issued by CA X for CA Y. CA Y issued CA-certificate 2 for CA Z, which has issued an end-user certificate for Bob’s key (certificate 3).

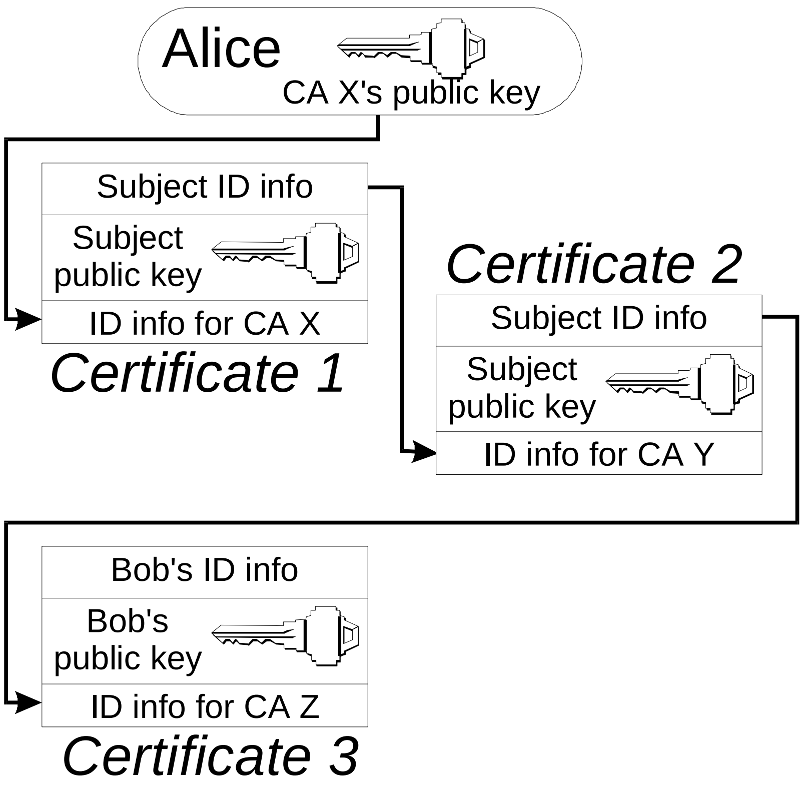

Figure 5 – A certification path from Alice to Bob

When Alice validates the certification path, she starts with CA X’s public key, which she uses to validate certificate 1. Then she uses the public key for CA Y she obtained from certificate 1 to validate certificate 2, thus acquiring CA Z’s public key, which she can then use to validate certificate 3 and securely obtain Bob’s public key.

How the CAs of a PKI are arranged is a basic PKI characteristic. Some PKIs use a general hierarchy, illustrated in Figure 66. In this picture, the circles represent CAs and the rectangles represent end users. An arrow indicates that the

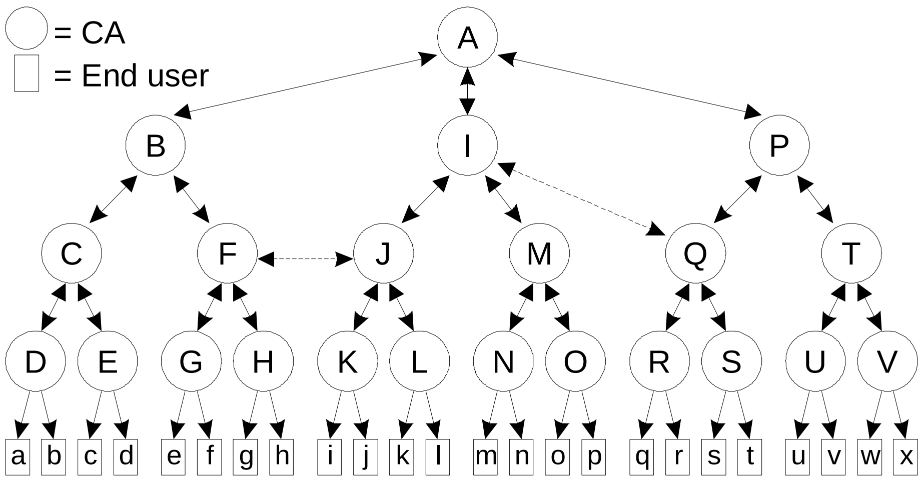

Figure 6 – A general hierarchy with cross-certificates

source has issued a certificate for the target. In a general hierarchy, each CA certifies its parent and its children. Also shown in Figure 66 are some *cross-certificates*, indicated by the dashed arrows, which are certificates that do not follow the basic hierarchy.

Some PKIs use a variant of the general hierarchy known as a top-down hierarchy, shown in Figure 77, in which CAs only certify their children and the top-level CA is the source of all certification paths.[7] Still other PKIs have no structure at all – in effect, each CA is its own root CA and has full authority over how its trust is assigned. These unstructured PKIs can operate in many ways. For example, a program called Pretty Good Privacy uses an unstructured PKI in which each CA bases its trust on the certificates of other CAs. If enough of the other CAs issue certificates that bind a particular name to a particular key, then the CA can accept that binind itself with some confidence. This is called a *web of trust*. Other unstructured PKIs operate differently.

The CA relationships of a PKI govern its *scalability*. For a PKI to operate globally, its functions must scale up to billions of users while retaining its practicality: certification paths must be easily discovered and should not grow too long. For

[7] The source CA of a certification path is also called the *root* CA. This can cause confusion when discussing treelike CA organizations. We use the term *root CA* to indicate the source of a certification path, and *top-level CA* to indicate the CA that is the root of a treelike structure.

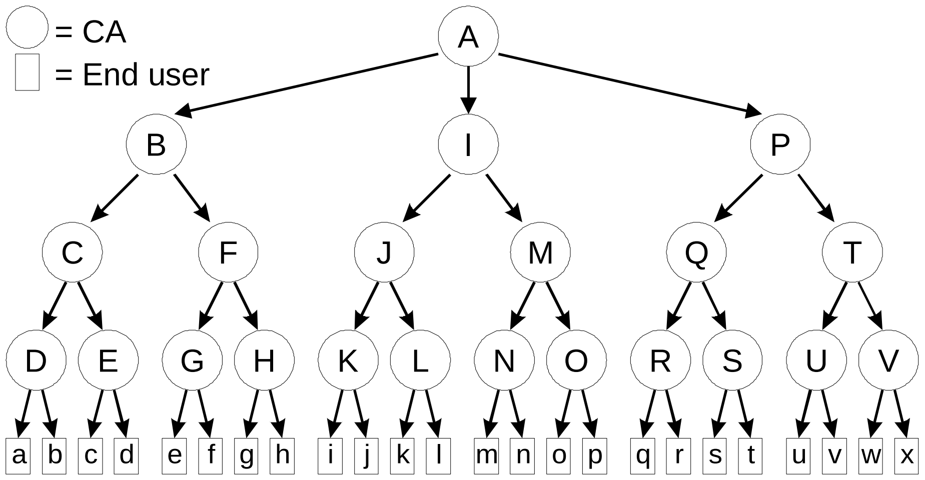

Figure 7 – Top-down hierarchy

example, a simple web of trust does not scale well. Although it is estimated that in a global web the average certification path length would be between 6 and 7 ([McBu]), path discovery can be difficult in a free-form web. There is also the problem of how trust is delegated. Even though Alice may trust Bob, when Bob trusts Carl, Bob may define his trust differently than does Alice.[8] So any keys that Alice receives from Carl (through Bob) may turn out to be unacceptable to Alice even though Bob may find them perfectly suitable. This forces Alice to no longer trust Bob, not because Bob is malicious in any way but simply because the two define their trust differently. This problem becomes worse as certification paths grow.

A hierarchical model scales well, but poses other problems. In a top-down hierarchy, all users must use the top-level CA as their root CA. This requires all users to obtain a copy of the top-level CA's public key *prior* to using the PKI. Also, all users must fully trust the top-level CA for all purposes. This makes a top-down hierarchy impractical for a worldwide PKI.

A general hierarchy lets any CA be the root CA of a certification path. However, the structure still relies heavily on the upper-level CAs, especially the top-

[8] Remember that CAs and end users need not be separate entities. In one PKI, that used by Pretty Good Privacy, all end users are in fact CAs.

level CA (CA A in Figure 66). A large number of certification paths in a general hierarchy pass through A. This forces the PKI's users to trust A implicitly. If A's private key were ever compromised it could be used to forge messages between entities which rely on a certification path that includes A. Since so many paths do pass through A, it becomes a very tempting target for attacks.

Certification paths in a general hierarchy also run the risk of becoming too long, resulting in problems similar to the web-of-trust. Cross-certification helps to reduce path lengths, at the risk of complicating path discovery. For example, in Figure 66 when user *h* is communicating with user *j*, should she take advantage of the cross-certificate between F and J, making her certification path G-F-J-K? Or should she follow the hierarchical path of G-F-B-A-I-J-K? It depends on how much trust she places in the different CAs in both paths, and on whether she knows about the cross-certification in the first place.

Entity Relationships

Two more basic characteristics stem from the relationships between a PKI's CAs, subjects and users. The first has to do with the kind of relationship that exists between the three – whether, for example, the CAs are distinct from the subjects and users or if users and subjects can also be CAs themselves. Also, when these three entities are distinct, the question arises as to how well they know each other and what minimum amount of familiarity is required for the PKI to operate. For example, is it necessary that a subject's CA also be the subject's employer? Or perhaps subjects and users have to know each other well enough to have at least met face-to-face.

The second relationship-based characteristic is the amount of trust that has to exist between the distinct entities of a PKI. Some PKIs require that users place, or *delegate,* all their trust in a single CA (for example, the top-level CA of a top-down hierarchy) while others allow users to decide which CAs to trust. Sometimes it is the CAs who must place their trust in other CAs or in their own subjects. A few PKIs allow their entities to refine the kind of trust they delegate. A CA could refine its trust in another CA such that the second CA would only be trusted for certain kinds of certifications. For example, the first CA could have a policy stating “I only trust this other CA to issue certificates that relate an email address to a public-key value” and this could be expressed in the PKI in a way that makes conformance to the policy automatic.

When dealing with trust issues the subject of liability inevitably arises. Who assumes responsibility for what can become very important when a PKI is used to secure sensitive information. In some PKIs, trust relationships are explicit and easy to audit, making it relatively simple to assign responsibility. Other PKIs have few, if any, mechanisms for encapsulating trust, let alone how it gets delegated. These PKIs make liability difficult to determine.

Trust issues are extremely complex and cannot be resolved by good PKI design alone. Trust is rarely defined in absolute terms. A person usually trusts somebody else for some things but not for others. No pre-defined model can hope to encompass all of the legal and social ramifications of trust. A PKI is merely a tool for expressing trust relationships. Any PKI that seeks to do more inevitably suffers from a lack of flexibility.

Validation

The second basic PKI operation is certificate validation. The information in a certificate can change over time. A certificate user needs to be sure that the certificate’s data is true – we say that the user needs to *validate* the certificate. There are two basic methods of certificate validation:

- The user can ask the CA directly about a certificate’s validity every time it is used. This is known as *online* validation.
- The CA can include a *validity period* in the certificate – a pair of dates that define a range during which the information in the certificate can be considered as valid. This is known as *offline* validation.

A PKI can use either or both methods. How a certificate user validates certificates is a basic PKI characteristic.

Closely related to the validation method is certificate *revocation*. Certificate revocation is the process of letting users know when the information in a certificate becomes unexpectedly invalid. This can occur when a subject's private key becomes compromised, or, more benignly, when a certificate's identifying information changes (e.g. the subject gets a new telephone number).

If a certificate is validated online with the CA every time it is used then the revocation problem becomes trivial, as the CA can simply state that the certificate is no longer valid. However, when validity periods are employed, the certificate revocation method becomes critical (especially in the case of private-key compromise). How a PKI revokes certificates is a basic PKI characteristic.

In the absence of online approaches, the most common revocation method uses *certificate revocation lists* (CRLs). A CRL is a list of revoked certificates that is signed and periodically issued by a CA. It is essential that the user check the latest CRL during validation to make sure that a certificate she is about to use has not been revoked.

One of the chief concerns with the CRL approach is what happens between the time when a CA receives notification that a certificate should be revoked and when the CA publishes its next CRL. Since the revoked certificate will only appear on the next CRL, any user checking the current CRL will not know of its revocation and will assume that the certificate is still valid. We call this the *CRL time-granularity problem.*

Another concern is the size of the CRL. A CA can be expected to certify thousands, or even hundreds of thousands, of subjects. While the rate of revocations for a given population is generally unpredictable, the CRLs for such CAs can be expected to grow very large. When a CRL is too large it can be difficult to retrieve by users, whose access to the CA may have limited bandwidth. Also, since CRLs are signed, their signatures need to be verified before the CRL can be used, and the time required to verify the signature on a large CRL and process its entries can become significant.

These problems have lead to several refinements of the CRL approach. One is to issue separate CRLs for different revocation reasons and/or for different certificate subjects. For example, the CA could issue one CRL for routine revocations (e.g. a change in a certificate subject’s identifying information) and another CRL for revocations due to a security compromise. Similarly, a CA could issue one CRL for its end-user subjects and another for the other CAs it may certify. These measures have the effect of partitioning a large CRL into pieces that can be selectively digested. For example, a user might not be very worried about routine revocations and so would only need to check the security-compromise CRL. Also, when processing a certification path the user need only check the CA CRLs (until he reaches the end of the path).

While these steps help reduce CRL sizes, they do little to alleviate the CRL time-granularity problem. Another measure has been proposed to address that problem: *delta-CRLs*. A delta-CRL is simply a (CA-signed) list of CRL changes that have occurred since the last full CRL was issued. Delta-CRLs allow revocation notifications to be issued more frequently, and so reduce the probability that a revoked certificate will be falsely validated. Delta-CRLs also help with the CRL size problem. A certificate validating system could start with a full CRL, and then need only process delta-CRLs as they are issued, updating its own copy of the full CRL. A complete discussion of CRL issues can be found in [FoBa].

Online revocation and validation methods are still very new. While it appears that an online approach avoids CRL management problems, the bandwidth and processing requirements of such approaches remain unclear.

Authentication

Authentication is the process of using a PKI. When a CA certifies an entity and a user then validates that certification, the entity is said to have been *authenti-*

cated. The degree to which the user can trust the certificate’s information and its validity is a measure of the *strength* of the authentication. For example, if you look at Alice and see that her eyes are blue, then you have a very strong authentication of the colour of Alice’s eyes (well, at least to the extent that you can trust your senses). On the other hand, if someone who has never seen Alice tells you that they have heard that Alice’s eyes are, say, “bluer than sunrise on Jupiter” then you really have no knowledge of Alice’s eye colour at all, since this information came to you very indirectly and since, for all anyone knows, Jovian sunrises may in fact be red. This is a very weak authentication.

As we mentioned under **Certification**, a certificate can contain entity or non-entity information. When a certificate identifies an entity, it is called an *identity certificate*. Authenticating an identity certificate is called *identity authentication*.

Certificates that contain non-entity information, such as a permission or credential, are called *attribute certificates*. In this thesis we will instead use the term *credential certificates* to avoid confusion with ANSI draft standard X9.45, which deals with attribute certificates in a specific way. Credential certificates identify things such as permissions (e.g. “can access computer `xyz`”), credentials (e.g. “is a certified stock broker”), or other attributes (e.g. “is VP Marketing for ABC Inc.”). A credential certificate may or may not identify the entity to which the credential is attached. We call authenticating a credential certificate *credential authentication*.

Whether a PKI uses identity or credential certificates, or both, is a basic PKI characteristic.

Limitations of PKI Authentication

Whenever authentication is performed using the PKI, whether online or offline, it is called *in-band authentication*. Authentication performed using more traditional methods, such as over the telephone or physically meeting someone, is called *out-of-band authentication*. The goal of every PKI is to minimize the need for out-of-band authentication, and its success in this endeavor is a basic PKI characteristic.

It is unlikely that out-of-band authentication can ever be completely eliminated. At the very least, a person wishing to use a PKI needs to first have their identity and/or credentials verified by their CA. This initial verification can not be performed using the PKI, since there is no other CA to vouch for the person's identity/credentials. Thus the bootstrapping process requires out-of-band authentication. Also, different PKIs require different degrees of out-of-band authentication as identity and credential information changes over time and needs to be updated.

The extent to which out-of-band authentication is required in a PKI is partly a result of how much the PKI's designers want to provide *irrefutability*. A signature made by Alice is said to be irrefutable if Alice can not, at a later date, deny that she did in fact make the signature. If the PKI is to be used as the foundation of an electronic replacement for paper-based signatures, then irrefutability is an important consideration. In general, the more out-of-band contact Alice has with her CA, the less she will be able to engage in such fraud.

This issue has legal and social, as well as technical, implications, and a detailed discussion is beyond the scope of this thesis.[9] While it does seem likely that PKIs can provide a better authentication system than the non-cryptographic ones which are currently used, many issues need to be resolved before that can happen. For example, what does "better" really mean? Public-key cryptography and PKIs provide an alternative method of authentication, but whether it is stronger or merely more convenient remains to be seen.

The degree of irrefutability provided by a PKI is certainly a basic characteristic. However, it depends on many non-technical factors such as the legal and social framework in which the PKI operates. There are also many technical factors outside a PKI's realm of control that impact irrefutability, such as how entities manage their private keys. For this thesis, we will concentrate on the technical aspects of irrefutability that can be provided by a PKI, and will briefly mention the other considerations only when they are influenced by a specific technical feature.

Anonymity

The degree of irrefutability accorded by a PKI brings up another issue: Anonymity, another basic PKI characteristic. We define anonymity as the ability to use the PKI while only revealing the information which is pertinent to the situation. An irrefutable signature seems to imply that the signer should be readily identifiable. Yet there are many situations where anonymity would be preferred, most notably in the area of shopping. For example, imagine a PKI that is set up to identify people by their name, address, phone number, place of work and job title. Such a PKI would be perfectly suitable when its users are acting in the capacities of their jobs. The CEO of a company would like to by identified as such when authorizing, say, a merger or a stock split.

However, that same CEO might be reluctant to use this PKI to make routine purchases, as her identifying information would be made available to whatever merchant she dealt with, information that she might prefer to keep private and/or that the merchant would like to use for marketing or perhaps even more nefarious schemes.

Ideally, a PKI should provide both strong, irrefutable authentication and a high degree of privacy through anonymity. Credential authentication holds the promise of giving a PKI those traits. As we shall see, several new PKI proposals use credential authentication to that end.

Summary

In this chapter we have described PKI operations and attributes in general terms, and identified ten basic PKI characteristics, which are set out in the following table. These characteristics will be used in the following chapters to describe current and proposed PKI systems.

Basic PKI Characteristics

9 See [Fr96] for a thorough review of digital signatures in a legal context. As an example of a social implication, consider the need to keep private keys secure, possibly through the widespread use of smart cards (see [Fa96]).

Certificate information	What data is contained in the certificate? Is it predefined, or can the certificate hold any kind of information?
CA arrangement	Are the CAs arranged in a strict hierarchy, or is the PKI unstructured? If unstructured, does the PKI use a web of trust or some other mechanism?
CA « Subject « User relationship	Are these three distinct entities? How well does each know the other? For example, is the subject an employee of the CA? Is that a requirement?
CA « Subject « User trust relationships	Where the entities are distinct, what is the degree of trust that they must place in each other? Who assumes the most liability in different situations?
Certificate validation method	Are certificates validated online, every time they are used or does the certificate contain a validity period, or both? If online, how are bandwidth issues addressed? If offline, how are CRL issues addressed?
Certificate revocation method	Is the revocation status of a certificate provided online or via a CRL? If CRLs are used, how are size issues and the time-granularity problem addressed? How do the validity periods of the CRLs relate to the validity periods of their certificates?
Identity vs. credential certificates	Does the PKI serve only as a means of identifying the public keys of entities, or can it also be used for credentials such as authorizations, permissions and other non-entity attributes?
Irrefutability and strong authentication	Do the users of the PKI have a way of showing that a signature was indeed generated by a particular subject?
In-band vs. out-of-band authentication	How much out-of-band authentication is required for the operation of the PKI?
Anonymity	Does the PKI provide its users with any anonymity? Are irrefutability and strong authentication diminished?

Table 1 – Basic PKI Characteristics

Chapter 4

A WORLD WITHOUT PUBLIC-KEY INFRASTRUCTURES

The Internet currently operates without the benefit of a public-key infrastructure. In this chapter we describe the present state of affairs for two common uses of the Internet that could be among the first to benefit from a PKI: Email exchange and FTP (File Transfer Protocol) access. We will return to these examples in later chapters to show how they can be enhanced by the various PKIs examined. The reader is assumed to have a basic familiarity with the operation of the Internet, email and FTP.

Insecure Email

People exchanging email over the Internet are exposed to two kinds of security risks. They can neither protect their messages from being read (or intercepted and even changed) by a third party, nor can they be assured that the person they are communicating with is in fact who they believe it to be.

As email is forwarded through the network, it passes through various computers such as routers and email servers. There is currently no mechanism that prevents the administrators of these machines from reading, copying and even modifying a message as it passes by. They can easily do so without being detected by the sender and receiver of the email. The message can even be intercepted and replied to, with the interceptor masquerading as the recipient. Only careful examination of the email message's headers can reveal this deception.

Furthermore, when Alice obtains Bob's email address, she usually has very little assurance as to it's authenticity. She can be fairly certain that it is valid if she obtained it directly from Bob, perhaps over the phone or if they exchanged business cards. She may even know the address via some close, mutual friend. These cases, however, are quite rare. Much of our knowledge about the origins of the Internet email addresses we use comes from the Internet itself, and as such is subject to the same security problems described above.

Figure 88 illustrates these problems. Before Alice can send email to Bob, she must obtain Bob's address. This is indicated in the figure by the black arrows, where Bob either gives Alice his address via some out-of-band means, or he sends it to her over the Internet. Note that any Internet communication between Bob and Alice passes through Oscar. If Bob elects to give Alice his address using the Internet, Oscar may intercept the message, substitute a different email address for Bob's and present himself as Bob to Alice.

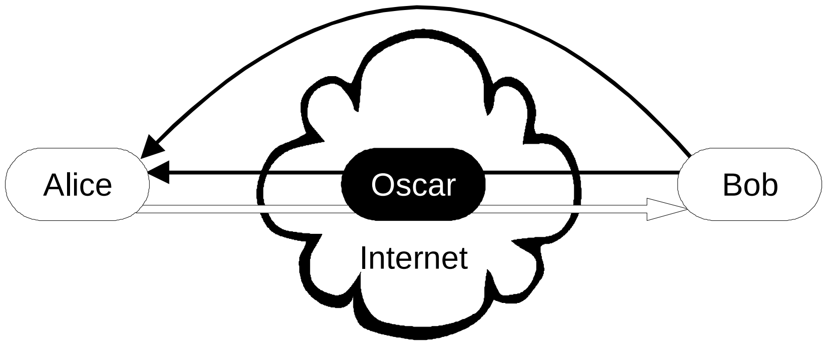

Figure 8 – Internet communications

Once Alice has obtained what she believes to be Bob's email address, she can send a message to Bob as indicated by the white arrow in the figure. Once again, Oscar is in a position to intercept the message. He may read it or modify it. If Alice and Bob used only the Internet to communicate with each other, then neither would be able to detect or prevent Oscar's meddling.

Despite these problems, Internet email has become the single most popular use of the network, with relatively little concern for security. This can be attributed to the facts that most Internet email traffic is not of a sensitive nature[10] and that the sheer volume of email makes it costly to find those messages which may be of interest. However, as the Internet becomes more mainstream the inadequacies of the current email system are becoming more apparent. A secure system is re-

[10] For example, most lawyers do not use Internet email to exchange case information, as they would be too exposed to betraying their client's confidentiality.

quired before people can use the network to exchange more sensitive information.

FTP and Access Control

Another common use of the Internet is to publish files via an FTP server or, more recently, a World-Wide Web server. Here problems arise when one does not want to make the files accessible to the entire Internet community. In general, the problem is one of access control, where access to a resource (such as a file) is restricted to a select group.

Currently such controls are provided by password-protecting the resource. Even if we disregard the possibility of a password being read by a third party as it is transmitted over the network, we still have problems administering a password-based system. The password has to be communicated securely to each user. In the current Internet this means using an out-of-band (non-Internet) medium. If the password is ever changed, each user must again be securely contacted.

This situation also leads to users having to remember many passwords, at most one for each resource they access. And, of course, we can not disregard the fact that passwords are transmitted “in the clear” and may be copied en route. Since resource access is usually performed interactively, it is more difficult for a third party to intercept and modify messages on-the-fly as with email. However it is not impossible, and there is still no way to prevent someone who knows the password from accessing the resource.

Figure 99 illustrates this simple access control mechanism for the FTP protocol. Bob must first communicate the password to Alice (the dashed arrow in the figure), hopefully using a more secure medium than the Internet. Alice then uses the password to access Bob’s FTP server (black arrow) and start an FTP session (white arrow). Oscar can obtain the password, since all Internet communication between Alice and Bob passes through him. He can then access Bob’s FTP server as Alice. Even without the password, he could masquerade as Bob’s FTP server to Alice.

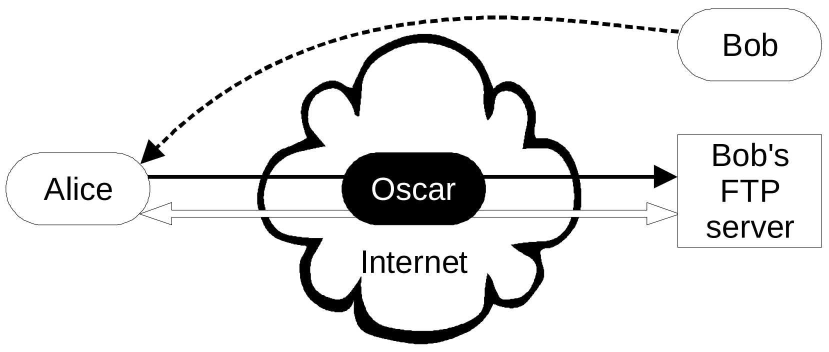

Figure 9 – The FTP protocol

Cryptography promises to secure Internet communications. A well-designed public-key infrastructure will minimize the problems associated with administering the cryptographic keys, and make those problems more manageable. The basic characteristics described in Chapter 3 capture the essence of a public-key infrastructure, and help to expose the strengths and weaknesses of a PKI. We will use the email and FTP examples in the following chapters to illustrate the link between an infrastructure’s characteristics and its effects on real systems.

PRETTY GOOD PRIVACY

Introduction

Pretty Good Privacy (PGP) is a public-key cryptography program created by Phil Zimmermann ([Zimm]). It uses RSA and IDEA (a symmetric key cipher similar to DES) to encrypt and/or sign messages, and was originally designed for use with Internet email. PGP users each maintain their own list, called a *keyring,* of the public keys of the people with whom they correspond. As a precaution against malicious tampering, the keyring is signed by the user's own private key, and so when she adds a key to her keyring she is said to have *signed the key.*

PGP allows users to exchange keyrings. When Alice adds Bob's key to her keyring, she assigns his key one of four attributes:

- Completely trusted – if any other key is signed by this key, then add the new key to the keyring. In effect, Alice is saying she trusts Bob to vouch for the validity of any key.
- Marginally trusted – a key signed by this key must also be signed by one (or more) other keys before it is added to the keyring. That is, Alice does not trust Bob very much, and needs to have his claims about keys corroborated by one or more others.
- Untrusted – do not use this key in determining whether other keys can be added to the keyring. Alice does not trust Bob to vouch for any keys at all.
- Unknown – a level of trust can not be determined for this key. In practice, this is the same as designating it as Untrusted.

These attributes allow Alice to assign a level of trust to Bob's key. Alice can tune PGP's criteria for accepting a key. For example, she can tell PGP to accept a key if it has been signed by two completely trusted keys or at least three marginally trusted keys.

PGP in Action

Since PGP is designed to secure Internet email, we will use email to show how PGP works. Figure 1010 depicts a simple PGP scenario. The dashed arrows represent out-of-band transmissions of a public-key. In our example, Alice and Bob met one day and Bob gave Alive his public key. At some other time, Bob met Chris and they exchanged their keys. And on yet another occasion, Chris met Elvis (at the supermarket, perhaps) and obtained his public key.[11]

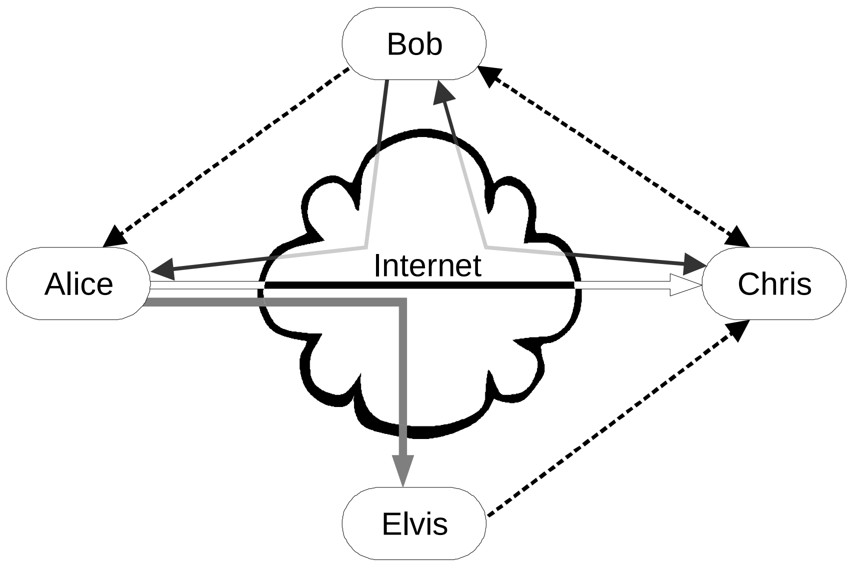

Figure 10 – Pretty Good Privacy

At this point, Bob's keyring contains keys for Alice and Chris, and Chris's keyring contains keys for Bob and Elvis. Bob now decides to exchange keyrings with Alice and Chris. This is illustrated by the solid black arrows in Figure 1010. Note that all Internet communications in this example are via email. Bob first contacts Chris and they exchange keyrings. Since Bob and Chris have both signed their

[11] We assume that each user fully trusts the others they meet outside of the Internet, since involving PGP's marginal trust capability would only complicate the example without contributing to it. The issues we raise here are not prevented or diluted through the use of marginal trust.

keyrings, and since they both know each other’s public key, this exchange can occur securely over the Internet.

Since Bob does not have Alice’s public key, he can’t obtain her keyring, but he can send her his keyring securely since he signs it and Alice knows his public key. In this way, Alice obtains the keys for Chris and Elvis.

The limitations of PGP’s web of trust model now become clear. Alice can use Chris’s key to communicate securely with Chris (white arrow in Figure 1010), and can be reasonably assured of doing so since Bob obtained Chris’s key directly before giving it to Alice. However, when Alice emails Elvis (thick gray arrow) she is in fact relying on the word of someone she never met (Chris) to tell her the key of someone else she never met (Elvis). If Alice had met Chris, perhaps she would not place that much trust in him. Even in that case, because she trusts Bob she winds up trusting Chris. Alice has no way of knowing which keys came from Chris unless Bob tells her explicitly.

In the end, if Alice trusts any one person in the web she must trust the entire web. And even if Alice finds that acceptable she is still vulnerable. Perhaps the Elvis that Chris met was in fact an Elvis impersonator. If Chris is fooled, then everyone who relied on Chris to obtain Elvis’s key is also fooled.

The Pretty Good Privacy PKI

As users trade keyrings, they build up a web of trust. In many ways, this is the simplest form of PKI. Each user is, in effect, her own root CA with full authority over how she assigns her trust. This simplicity has allowed PGP to gain relatively widespread acceptance on the Internet compared to other PKIs. However, where electronic commerce and other applications that require strong authentication are concerned, the PGP PKI falls short.

A PGP certificate is not extensible and contains only an email address, a public-key value and a degree-of-trust attribute. Since an email address is by no means an accurate method of identifying someone, PGP can not provide strong authentication of a person’s identity. The certificate’s lack of extensibility prevents PGP from being used for applications beyond casual email communication. For example, a bank cannot create a PGP bank account certificate for Bob’s public key. Even if such a certificate were possible, PGP does not allow a user’s trust to be delegated in a refined fashion. A user signing Bob’s bank’s key has no way to say “this is Bob’s bank’s key,” let alone something like “I trust Bob’s bank only for Bob’s financial information.”

PGP does not have a coherent method for validating and revoking certificates. Both functions are performed essentially by word-of-mouth. In general, Alice will learn of Bob’s public key via the web of trust. She has no way of being sure that the information she receives is correct, or if it has been maliciously tampered with or even forged.

If Alice is sending an encrypted message to Bob and she wants to be absolutely sure that she is using Bob’s current public key then she must communicate with Bob via some out-of-band means (e.g. a telephone call) prior to sending her message. She must do this for each message she wants to be absolutely sure of encrypting properly. Conversely, if Bob wants to be sure that Alice knows that he has changed his public key, he must tell her directly, also via some out-of-band method. He can’t just send Alice a message, signed with his old key, telling her the value of the new key. If the old key was compromised, such a message should not be trusted. And since Alice has no way of knowing if Bob’s key was compromised or not, she should never trust such a message. Bob could use the old key to securely inform Alice of the key’s compromise (once Alice knows of the compromise, she simply stops trusting the old key) but he can not then use the old key to convey the value of the new key, and so must send the new key to Alice directly. This need for direct, out-of-band communication in order to obtain strong authentication greatly hampers the use of PGP for anything more than casual email communication.

PGP’s use of email addresses as its sole means of identifying subjects also prevents its users from having any degree of anonymity. A subject could use a “false” email address, one which gives no indication of the true identity of the person behind it. However this would destroy any chance for any reliable authentication beyond the online persona presented by the email address.

The PGP PKI is simple and has gained widespread acceptance, but it is unsuitable for most applications beyond casual communication. Table 22 summarizes the PGP PKI in terms of the basic PKI characteristics.

PGP PKI Characteristics	
Certificate information	The PGP certificate is simple and rigid. It contains only a public key, an email address, and the degree-of-trust attribute. It is not extensible.
CA arrangement	PGP CAs are arranged in a web of trust.
CA « Subject « User relationship	Each PGP user is her own root CA. Subjects may or may not be CAs.
CA « Subject « User trust relationships	Since each user is their own CA, the PGP user completely trusts her CA. The CAs can assign a degree of trust to their subjects (i.e. other CAs), but they have no way of preventing their trust from being infinitely extended.
Certificate validation method	PGP uses neither online validation nor validity periods. Once a certificate is added to a user’s keyring, it is considered valid until the user decides otherwise.
Certificate revocation method	PGP relies on word-of-mouth to propagate information about revoked certificates. PGP does not use CRLs.
Identity vs. credential certificates	PGP uses purely identity certificates. They have no provisions to include credentials.

Irrefutability and strong authentica-tion	PGP has very weak authentication. The sole means of identifying a subject is with an Internet email address.
In-band vs. out-of-band authentica-tion	PGP relies almost entirely on out-of-band authentication.
Anonymity	PGP does not provide for any direct anonymity. A degree of ano-nymity can be achieved by using a “fake” email address.

Table 2 – Basic Characteristics of the PGP PKI

Chapter 6

X.509

Introduction

X.509 is the authentication framework designed to support X.500 directory services. Both X.509 and X.500 are part of the X series of international standards proposed by the ISO and ITU. The X.500 standard is designed to provide directory services on large computer networks. X.509 provides a PKI framework for authenticating X.500 services.

The first version of X.509 appeared in 1988, making it the oldest proposal for a worldwide PKI. This, coupled with its ISO/ITU origin, has made X.509 the most widely adapted PKI. There are at least a dozen companies worldwide that produce X.509-based products, and that number is growing. Visa and MasterCard have adapted X.509 as the basis for their Secure Electronic Transaction standard ([SET]). Netscape's famous World Wide Web software also uses X.509. And there are numerous X.509-based products available from companies such as Entrust and TimeStep that support corporate "intranets." Efforts are currently underway to design an X.509-based PKI that will support a global network such as the Internet. Along with PGP, X.509 is the only PKI system that has yet to be put into practical use.

Version 3 of X.509 is currently in the final stages of adoption by the ISO and ITU. X.509v3, as it's called, greatly extends the functionality of the X.509 standard. Most products available today use version 1 or 2 of X.509, with only a few working systems based on version 3. The SET protocol is based on version 3, as are most of the proposals for a global X.509 PKI.

In this chapter we will describe all versions of the X.509 standard, highlighting its general strengths and weaknesses. We will then describe two X.509-based Internet PKI proposals: the failed PEM and the draft PKIX standards.

The X.509 Standard

X.500

A full understanding of X.509 PKIs requires some basic knowledge of the X.500 directory that X.509 was originally designed for. The X.500 directory is very similar to a telephone directory where, given a person’s name, one can find auxiliary information about that person. However, X.500 provides more than just a name, address and phone number. An entry in an X.500 directory can contain a host of attributes, such as the name of the organization the person works for, her job title and her email address, to name a few. An X.500 directory entry can represent any real-world entity, not just people but also computers, printers, companies, governments, and nations. The entry can also contain the certificate specifying the entity’s public key.

To support looking up entries in the directory, each entry is assigned a globally unique name, called a *distinguished name* or DN. To help ensure their uniqueness, these names are assigned in a very specific fashion. The X.500 directory is arranged in a hierarchical fashion, call the *Directory Information Tree* or DIT (see Figure 1111).

Each node, or *vertex*, in the tree has one parent (except the root vertex) and any number of children. Each vertex, except the root, is assigned a *relative distinguished name,* or RDN, that is unique amongst all the vertex's siblings. The RDNs of each of the vertex's ancestors are concatenated with the vertex's own RDN to form the entry's DN. Figure 1111 illustrates this process. Under the root vertex there is an entry for each country in the world. These entries are assigned an RDN that is the country's unique two-letter code assigned to it by the ISO. Beneath each country's vertex are entries for all of the country's organizations, such as it's government, its states or provinces, and federally-chartered companies. Each of these is assigned a unique RDN that is the name of the organization. Finally, each organization creates entries for all of its employees, and for other enti-

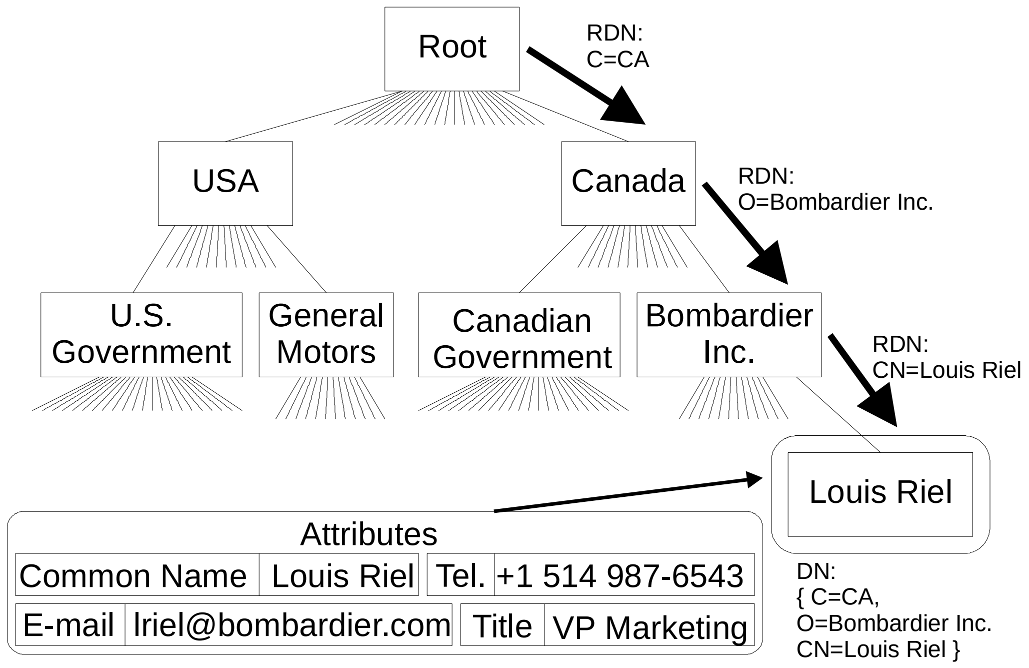

Figure 11 – The X.500 Directory Information Tree

ties the organization might control. Each of these is also assigned a unique RDN. In our example, Mr. Louis Riel works for Bombardier, a Canadian company. Bombardier assigns an RDN to Mr. Riel that is simply his name (specified as his "common name," abbreviated as CN in the figure). Bombardier was itself assigned an RDN, "Organization=Bombardier" by Canada, designating it as the organization named Bombardier, and Canada's RDN is it's two-letter country code, "Country=CA." Mr. Riel's DN is thus the concatenation of these RDNs, starting from the root: Country=CA, Organization=Bombardier, CommonName=Louis Riel.

X.509 Versions 1 and 2

X.509 was created to support the authentication of the entries in an X.500 directory. The latest version, the third, has evolved beyond its X.500 roots. Version 3 will be officially adopted as the X.509 standard sometime in 1997. Currently, version 2 is the official standard. However, copies of the final draft of version 3 have been made available to help speed its adoption. We will first describe X.509v2, before moving on to the extensions added under version 3.

The X.509v2 certificate is illustrated in Figure 1212. It contains the following fields:

- Version: The X.509 version that the certificate conforms to.
- Serial number: A unique number assigned to the certificate by its issuing CA.
- CA signature algorithm: An identifier for the algorithm used by the CA to sign the certificate. Identifiers are

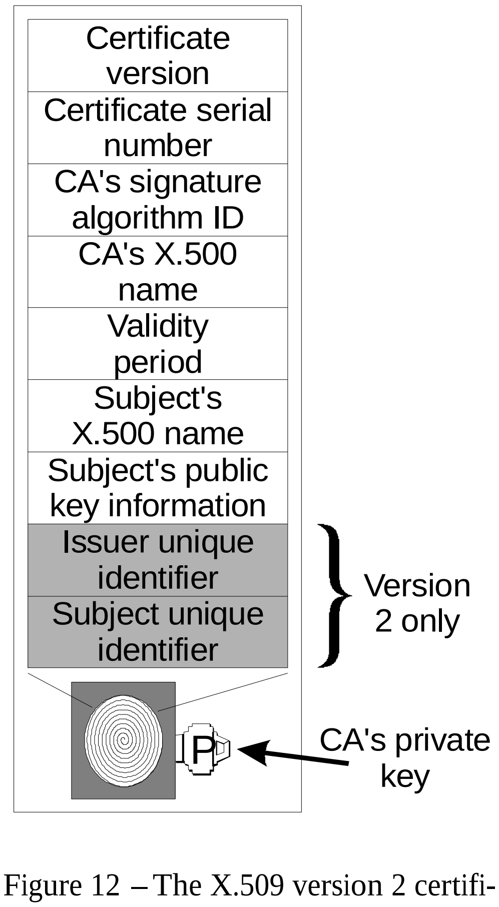

Figure 12 – The X.509 version 2 certificate

discussed further below under Object Registration.

- Issuer name: The X.500 name of the issuing CA.
- Validity period: A pair of dates / times between which the certificate is considered valid.
- Subject name: The X.500 name of the entity who holds the private key corresponding to the public key being certified.
- Subject public key information: The value of the subject's public key along with an identifier of the algorithm with which the key is intended to be used.
- Issuer unique identifier: (Optional, version 2 only.) A bit string used to make the X.500 name of the issuing CA unambiguous. It is possible for an X.500 name to be assigned to a particular entity, then de-assigned, then re-assigned to a new entity.[12] The unique identifier fields address this concern. These fields are not widely used, as they have turned out to be difficult to manage and are ignored or omitted in most implementations. The preferred method used to address this problem is to design the RDNs in such a way as to ensure that they are *never* reused. For example, rather than use just the CommonName attribute, a better form of RDN might use both the CommonName and an EmployeeNumber.
- Subject unique identifier: (Optional, version 2 only.) A bit string used to make the X.500 name of the subject unambiguous.

Because of X.509's close ties with X.500, its CAs are usually arranged in a hierarchy that closely follows the X.500 DIT. X.509 itself does not dictate a particular CA arrangement, however it does describe the general hierarchical model with cross-certificates, and encourages its use with X.509.

[12] For example, in Figure 1111 if Mr. Riel changes companies, his DN, in particular the Organization=Bombardier component, is no longer valid and so is de-assigned. Later, if another person named Louis Riel comes to work for Bombardier, he would be assigned the same DN as the first Louis Riel.

Version 3 of X.509 provides the means to move beyond the need for a hierarchy, however current X.509 products – most of which are based on versions 1 or 2 and are designed mainly for use within a single organization – typically rely on the existence of some form of hierarchy. Organizations such as a large company lend themselves well to a hierarchical model. The company would typically issue certificates for its employees, and the corporate PKI would only be used for internal communications. The PKI would follow the innate hierarchy of the company, making trust relationships easy to define and manage – the employee naturally trusts the company to issue correct certificates regarding other employees.

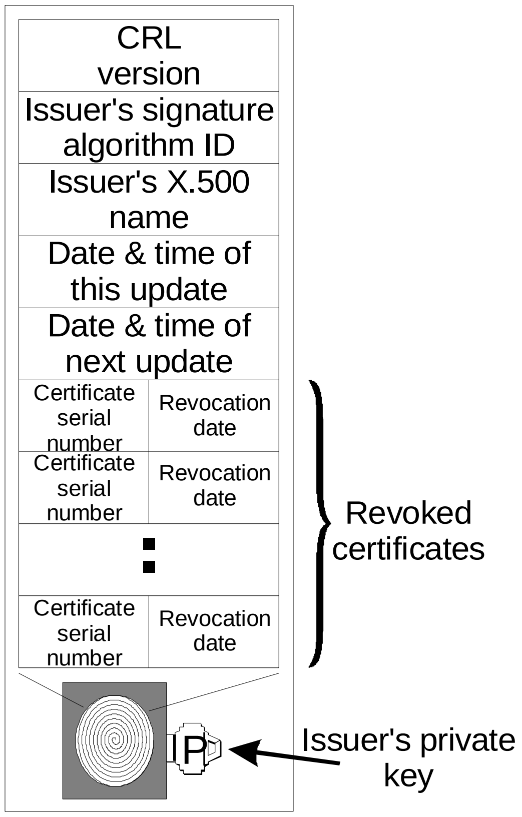
CRL version
Issuer's signature algorithm ID
Issuer's X.500 name
Date & time of this update
Date & time of next update
Certificate serial number
Revocation date
Certificate serial number
Revocation date
Certificate serial number
Revocation date
Revoked certificates
IP
Issuer's private key

Figure 13 – X.509v1 CRL format

X.509, and X.500, were originally designed in the mid-1980's, before the current explosive growth of the Internet. They were therefore designed to operate in an offline environment, where computers are only intermittently connected to each other. X.509 thus employs CRLs. Versions 1 and 2 of X.509 use very simple CRLs that do not address size issues and the time-granularity problem. Version 3 makes several attempts to solve these problems, with varying success. Figure 1313 illustrates the CRL format used in X.509 versions 1 and 2.

X.509 Version 3

Version 3 introduced significant changes to the X.509 standard.[13] The fundamental change was to make the certificate and CRL formats extensible. X.509

[13] X.509 version 3 is currently awaiting final approval and ratification by the ITU.

implementers can now define certificate contents as they see fit. Also, a number of "standard extensions" were defined to provide key and policy information, subject and issuer attributes, certification path constraints, and enhanced CRL functionality. These extensions are fully described in [FoBa] and elsewhere. We concentrate here on those extensions which affect the basic PKI characteristics of X.509.

Version 3 Certificate Extensions

Certificate policies and policy mapping. X.509v3 gives CAs the ability to include with the certificate a list of policies that were followed in creating the certificate. These policies are intended to help users decide if a certificate is suitable for a particular purpose. For example, a policy might indicate that a certified key can be used for casual email messages but not for financial transactions. In general, a certificate policy indicates such things as CA security procedures, subject identification measures, legal disclaimers or provisions, and others.[14] Policy mapping allows a CA to indicate whether one of its policies is equivalent to another CA's policy.

Alternative names. An X.509v3 certificate can contain one or more alternative names for the subject or issuer. This allows X.509 to operate without an underlying X.500 directory. Examples of alternative names include email addresses and World Wide Web universal resource locators. Implementers can also define their own alternative name forms. Alternative names can also be used to identify the issuer of a CRL.

Subject directory attributes. This extension allows any of the subject's X.500 directory entry attribute values to be included in the certificate. This allows the certificate to carry additional identifying information beyond the subject's name(s).

Certification path constraints. These extensions allow CAs to link up their infrastructures in meaningful ways. A CA can restrict the kinds of certification paths that can grow from certificates it issues for other CAs. The CA can state

[14] X.509v3 allows CAs to define any certification policy they require.

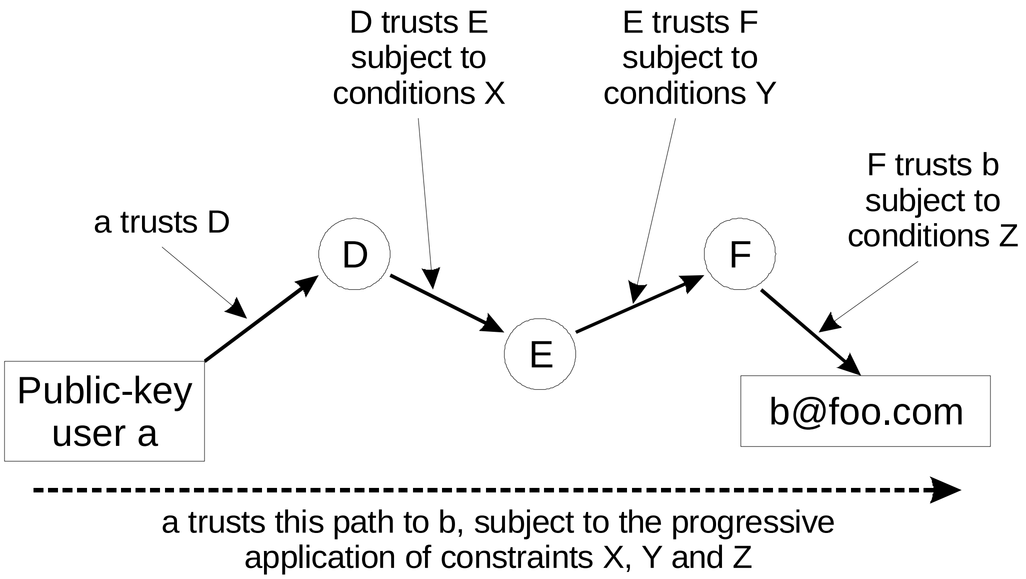
D trusts E subject to conditions X
E trusts F subject to conditions Y
F trusts b subject to conditions Z
a trusts D
D
E
F
Public-key user a
b@foo.com
a trusts this path to b, subject to the progressive application of constraints X, Y and Z

Figure 14 – A progressive-constrained trust chain

whether or not a certificate’s subject is in a fact a CA (to prevent an end user from fraudulently acting as a CA). The CA can also constrict paths growing from the certificate to certificates issued in a particular name space (e.g. within a given Internet domain) and/or to certificates that follow a specific set of certification policies. This is an important extension because it allows CAs to employ a *progressive-constraint* trust model that prevents the formation of infinite certification paths. The concept is illustrated in Figure 1414. User a uses D as her certification authority, so she places complete trust in D. D has certified another certification authority, E, for example only trusting E to issue certificates for other CAs (perhaps E performs some kind of national CA registration). Constraint X would then state that D only trusts E to certify other certification authorities.

E has issued a certificate for certification authority F stating that it only trusts F to issue certificates for end users in the domain `foo.com`. So constraint Y would state that E trusts certificates issued by F only if they certify an end user *and* that user’s name is in the `foo.com` domain. Finally, F issues a certificate for user b, but only trusts b for casual email (as opposed to, say, making financial commitments on F’s behalf). So constraint Z states that the certificate issued for b by F should only be used for casual email.

In this way the unlimited trust that a places in D becomes increasingly constrained as the certification path grows. When a obtains a certificate for b she knows that she should only use it for casual email, and she has greater confidence in the strength of the authentication than with, say, PGP’s web of trust because she can see how trust has been restricted along the certification path. Given these constraints, she would not accept a certificate issued by E for b (or any other user), nor would she accept any certificates from any certification authority certified by F. If CAs define the tightest practical conditions when they certify other CAs, then as a certification path grows it becomes progressively more constrained until it can grow no longer.

Version 3 CRL Extensions

CRL number and reason codes. Each CRL issued for a given certificate population is assigned a number from a monotonically increasing sequence. This allows users to determine if a CRL was missed. Also, each certificate in a CRL can now have a revocation reason attached to its CRL entry. These two extensions allow for the more sophisticated CRL extensions described below.

CRL distribution points. This extension helps reduce the sizes of CRLs processed by a CA’s users. Rather than forcing users to accept the full CRL, the CA can partition the CRL in some way, and issue each partition from a different distribution point. For example, a corporate CA might issue a different CRL for each division of the company. Then when a user wants to verify a certificate for someone from a particular division, she need only check that division’s CRL rather than the full CRL. Another way of partitioning the CRL is according to revocation reason. Routine revocations, for example, those due to a name change, can be placed on a different CRL than revocations due to a security compromise. The compromise list can then be updated and checked more frequently without having to also process all the routine revocations that might occur.

Delta-CRLs. This extension provides another method of reducing CRL sizes. Rather than issue a full CRL (or a full partition of a CRL), the CA can simply issue a list of the changes that have occurred since the last time a full CRL was issued. Users that maintain their own CRL database can use a delta-CRL to keep their copies updated without having to download and process all the entries of a full CRL, saving bandwidth and computing time.

Indirect CRLs. This extension allows a CRL to be issued from an entity other than the CA that issued its certificates. This allows for CRL "clearing houses" which would gather the CRLs from multiple CAs and provide one distribution point for them all.

All of these CRL extensions still do not overcome the fundamental time-granularity problem. Even with partitioned CRLs and frequent delta-CRL issuance, there is still a window of opportunity for a compromised certificate to be used. The X.509v3 framework can be used for online operation, avoiding the need for CRLs altogether. However, there is as yet no system defined for such use.

Object Registration

The extensibility of X.509v3 gives it a tremendous amount of flexibility. However, the way in which that extensibility is provided hampers the widespread application of user-defined extensions for a global PKI.

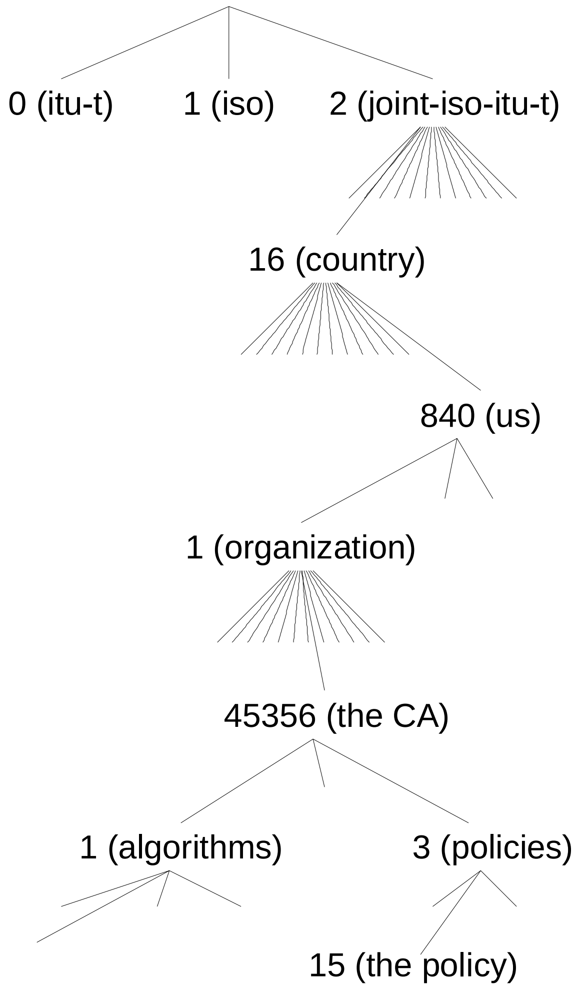

Figure 15 – Object registration example

Every time X.509 needs to identify some object – such as a signature algorithm, certification policy, user-defined alternative name or a user-defined extension – it uses an internationally defined *object identifier* mechanism. An object identifier, or OID, is a numeric value, composed of a sequence of integers, that is unique with respect to all other OIDs.

The OIDs are assigned following a hierarchical structure of value-assigning authorities (see [FoBa]). Essentially, any company or organization can become a value-assigning authority. The company is itself assigned a value that serves as a prefix for all the values that it defines.

Take, for example, the OID pictured in Figure 1515. Imagine a CA operating in the United States. The CA would be assigned an OID, say 2-16-840-1-45356.[15] This OID would then be the prefix used for the OIDs of any objects that the CA cares to register. The CA might want to register a particular certification policy, to which it has assigned a number, say 15, beneath the "policies" branch of its hierarchy (branch number 3, for example). Then the CA's policy could be identified as object number 2-16-840-1-45356-3-15.

This system works well for assigning numbers to objects, and it is used extensively in X.509. For example, if the CA in Figure 1515 were to use its policy in a certificate, that policy would be identified solely by its OID. Difficulty arises, however, when OIDs are used without prior agreement as to their meaning. If the CA in our example wants to use their policy in their certificates, they have to ensure that the meaning of the OID identifying their policy is known *a priori* by any entity wishing to use the certificate. Otherwise, when an ignorant entity encounters the value 2-16-840-1-45356-3-15 it will have no idea how to interpret the policy.

Confusion can also arise when the same object is assigned multiple OIDs. For example, imagine that two CAs have each assigned an OID to a particular signa-

[15] The numbers only have meaning within the hierarchy. The leading 2 indicates the branch of the hierarchy administered jointly by the ISO and ITU. The 16 is the number assigned to the branch used by national registration authorities. 840 is the country code for the U.S., whose national registration authority (ANSI) uses 1 as the prefix for all the organizations it registers. The 45356 is simply a number assigned to the CA by ANSI.

ture algorithm, such as SHA-with-RSA. As long as the CAs and their users don't interact there will be no problems. However, if a user from one CA tries to use the other CA's certificate, he won't recognize the second CA's OID for SHA-with-RSA, and might assume that he can't verify the signatures of the certificate's subject even though he may be perfectly capable of doing so. The problem is compounded if the two CAs ever try to link their infrastructures. Then the CAs must either let all their users know that the two OIDs are equivalent, or one CA (or both) has to change its OID and communicate that change to all its users.

The OID problem prevents X.509's extensibility from being used freely on a large scale, since whoever creates a new extension must ensure that the relevant OIDs are known by all parties concerned. There is at present no systematic method for determining the meaning of an OID. They are neither regularly published nor are they reliably listed in a central registry. The only way you can find out the meaning of an OID is to have the OID's creator tell it to you.

Basic PKI Characteristics of X.509

We now look at how X.509 fulfills the basic PKI characteristics. Table 33 describes all versions, allowing them to be easily compared.

X.509 PKI Characteristics		
	Versions 1 & 2	***Version 3***
Certificate information	X.500 names only. Includes CA & subject names, subject public key, and a validity period.	Fully extensible, can include any information.
CA arrangement	No mandated CA arrangement, however the general hierarchy with cross-certificates is encouraged. No trust constraint mechanisms.	Trust constraint mechanisms are provided. The general hierarchy with cross-certification is still encouraged.

CA « Subject « User relationship	CAs, subject and users are distinct.	
CA « Subject « User trust relationships	Each user is expected to fully trust at least one CA. CAs have no mechanism for manipulating their trust relationships with subjects and other CAs.	Each user is expected to fully trust at least one CA. CAs can constrain how their trust in subjects and other CAs is delegated.
Certificate validation method	Offline. Certificate chains are stored locally and / or transmitted with every message. Validation is performed by checking the validity period of each certificate and verifying that the certificate does not appear on the latest available CRL.	Offline, but can be online through yet-to-be-defined extensions.
Certificate revocation method	Simple CRLs only.	Sophisticated CRL mechanisms. Online methods can be defined via extensions.
Identity vs. credential certificates	Identity certificates only. Credentials may be attached to the named X.500 directory entry.	Mainly identity certificates. Certain standard extensions provide some credential-like functionality. Can be extended to provide full credential certification.
Irrefutability and strong authentication	Authentication strength based on the accuracy of X.500 entries. CA is responsible for issuing certificates that are not misleading.	CA is still responsible for certificate accuracy, but use of non-X.500 names may make this more difficult.

In-band vs. out-of-band authentica-tion	Users must obtain at least one CA key out-of-band. Also, the extensive use of OIDs requires out-of-band communication whenever a new extension is defined.	
Anonymity	Anonymous only to the degree that an X.500 entry can be anonymous.	Extensions can be used to provide fully anonymous service.

Table 3 – Basic Characteristics of the X.509 PKI

X.509 on the Internet

We now turn our attention to X.509-based proposals for an Internet PKI. We first present the Privacy Enhanced Mail PKI, which is based on X.509v1. We conclude this chapter with a very brief look at the X.509v3-based PKIX standard, which is still in draft form.

Privacy Enhanced Mail

Privacy Enhanced Mail (PEM) was proposed in early 1993 as an Internet standard for cryptography-enhanced email (see [RFC1421], [RFC1422], [RFC1423] and [RFC1424]). The intention was to endow Internet email with "confidentiality, authentication, message integrity assurance, and non-repudiation of origin" using public-key cryptography. To this end, [RFC1422] proposed an Internet PKI to support PEM. The standard never caught on in the Internet community for various reasons, one of which was that its proposed PKI model proved to be a poor fit to the Internet's peer-based structure.

The PEM PKI uses a top-down CA hierarchy (see Figure 1616). At its root is the Internet Policy Registration Authority (IPRA), which establishes the global certification policies for the entire Internet. The IPRA only certifies Policy Certification Authorities (PCAs). Each PCA establishes more specific certification policies and certifies the CAs of different organizations that follow the PCA's policies. It was expected that there would be "a relatively small number of PCAs, each with a substantively different policy, to facilitate user familiarity with the set of PCA policies." Below the PCAs, each CA can certify other CAs or PEM users.

This sort of CA arrangement works well if there is an underlying hierarchy amongst the entities. The U.S. Department of Defense, for example, successfully uses a top-down hierarchy for its public key infrastructure. The DoD is naturally structured this way, so the hierarchy works very well. The Internet, however, has few hierarchical aspects. Also, since X.509v1 provides no inherent mechanism for publishing certification policies, the policies of PCAs were to be distributed via Internet RFCs, and users were expected to be familiar with the policies of the various PCAs, as well as the CAs they deal with regularly.

Essentially, PEM was an attempt to apply X.509v1 directly to the Internet. Each entity (users, CAs, PCAs, and the IPRA) would be assigned an X.500 Distinguished Name. Elaborate measures were defined to ensure that every DN would be unique. Prior to certifying a CA, a PCA would query a database (maintained by the IPRA) to determine if the CA's DN would be unique. Also CAs, but not

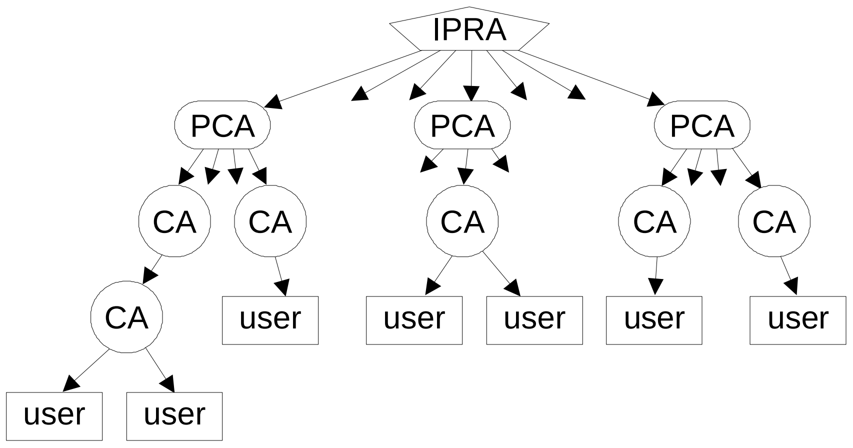

Figure 16 – The PEM CA hierarchy

PCAs, were to be subordinate to their DNs – they would only sign a certificate if the subject's DN was subordinate to the issuer's (CA's) DN.

The DN database was required because of the lack of a ubiquitous X.500 directory. Although no two PCAs, nor two CAs below the same PCA, would have the same DN, "since PCAs are expected to certify organizational CAs in widely disjoint portions of the directory namespace, and since X.500 directories are not ubiquitous, a facility is required for coordination among PCAs to ensure the uniqueness of CA DNs."[16] There were also special conditions under which multiple, distinct CAs might share the same DN and so a database would be needed to track them. For more details, see section 3.4.2.2 of [RFC1422].

PEM in Action

To illustrate some aspects of PEM's operation, assume that Alice wishes to send a message to Bob, with whom she has never communicated, and that they both operate within the certification structure shown in Figure 1717(a) (the arrows represent a certification, such as PCA1 issuing a certificate for CA1's public key). When Alice composes her message, she appends the certification path between her and the IPRA, i.e. the certificates for herself, CA2, CA1 and PCA1. Note that she does not issue (i.e. sign) these certificates, she merely includes them in her message (she

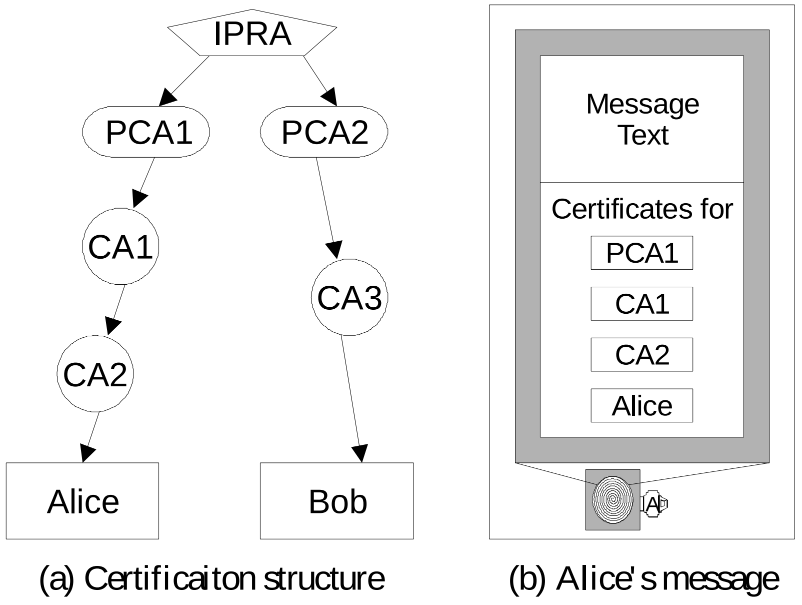

Figure 17 – PEM example

[16] [RFC1422], section 3.4.2.2 (page 14).

obtained them when she was issued her own certificate by CA2). She then signs the entire message, including the certification path (see Figure 1717(b)).

When Bob receives the message and wishes to validate it, he starts with the IPRA key and validates the certificate path included in the message to obtain Alice's public key (checking not only that the included certificates have neither expired nor been revoked, but also that Distinguished Name subordination has been observed). If this is the first time that Bob has followed a certification path down PCA1's branch of the hierarchy, his PEM software is required ask Bob to explicitly accept PCA1's certificate. This is to alert Bob that he is entering a new certification policy domain, and that he would be wise to review PCA1's policy (as well as the policies of any CAs subordinate to PCA1, depending on how much PCA1 allows its CAs to refine their own policies).

This requirement for policy awareness helps Bob to determine whether the origin of the message agrees with its contents. For example, Bob might be suspicious of a purchase order message requesting an educational discount that is certified beneath a PCA that represents commercial organizations. However, it is entirely up to Bob to be aware of the policies of the various PCAs. PEM has no automatic policy-verification mechanism. The need for all users to be familiar with the policies of each PCA they encounter resulted in PEM requiring that there only be a small number of PCAs. This meant that PEM would have had to describe every aspect of Internet communications with only a handful of policies.

The Privacy Enhanced Mail PKI

The cumbersome nature of its PKI contributed to PEM's downfall as an Internet standard. One of the primary lessons learned from experience with PEM is that a strict, hierarchical model does not work well in the global Internet. We now present the basic characteristics of the PEM PKI in Table 44.

The Privacy Enhanced Mail PKI Characteristics	
Certificate information	PEM certificates are X.509v1 certificates.

CA arrangement	PEM uses a rigid, top-down CA hierarchy.
CA « Subject « User relationship	Users and subjects are distinct from CAs. No PEM user can be a CA.
CA « Subject « User trust relationships	All PEM users must place complete trust in the IPRA, as all certification paths start with the IPRA’s key. A user must also trust the PCAs and CAs they encounter in a certification path. The user must be familiar with each PCA’s policies, and trust that the PCA and the CAs have not deviated from those policies.
Certificate validation method	Certificates are validated “online” using email. It is expected that the message originator would include the full certification path to his key in the message, which the recipient can validate using the IPRA’s key. While performing this validation, the user must also verify that no certificates have been revoked or expired, and that DN subordination has been followed.
Certificate revocation method	PEM uses X.509v1 CRLs, distributed via email to each user.
Identity vs. credential certificates	PEM certificates are purely X.509v1 identity certificates.
Irrefutability and strong authentication	The PEM architecture allows for strong authentication of users.
In-band vs. out-of-band authentication	Each user needs to obtain the IPRA’s key via some out-of-band means. Given that key, all other authentication can be performed in-band.
Anonymity	PEM provides an anonymity mechanism through what it calls “PERSONA” CAs. A PERSONA CA is distinct from a regular PEM CA in that it explicitly does not vouch for the identity of its subjects.

Table 4 – Basic Characteristics of the PEM PKI

Conclusions

Efforts are currently underway to define an X.509v3-based PKI for the Internet. Called PKIX (for, roughly, Public Key Infrastructure using X.509), the proposal is currently in an early draft stage. It closely follows the X.509v3 standard, with perhaps a few different extensions, and there is a strong push to have both standards agree as much as possible. In fact, many of the people involved in creating X.509v3 are also involved with PKIX. Although it is too early to speculate about the final nature of PKIX, we can assume that it will be very similar to an X.509v3 PKI.

The chief drawback to using X.509v3 is its dependence on object ID numbers. In the absence of a coherent method for disseminating the meanings of OIDs, care must be taken when using an X.509v3 PKI to ensure that every entity will understand all the OIDs it may encounter. PKIX has yet to address this issue. Some hope that a relatively small set of OIDs can be defined that would encompass enough practicality to make the PKI functional. Considering that X.509 is basically intended to be an identity PKI, with little or no attribute certification, it may be possible to create such a set. However it is far from certain that such a thing will happen even if it is possible. We note that the small set of policy definitions that PEM required for its PCAs was never realized.

Several people have found X.509 to be an unsuitable basis for a global PKI. This conclusion is based mainly on experience with X.509v2 and it’s dependence on X.500’s global name hierarchy. Many see the need to use a globally unique name as excessively complex for most situations, and question the importance of identity-based certificates. X.509 also has other deficiencies, such as its reliance on OIDs and a need to formally define every aspect of its operation. These issues have inspired people to “start over from scratch” in the hopes of creating a simpler system. We now turn our attention to some non-X.509 PKIs, starting with the proposed Secure DNS standard.

Chapter 7

INTERNET DOMAIN NAME SYSTEM SECURITY EXTENSIONS

Introduction

The Domain Name System (DNS) has become a critical part of the Internet's operations. The DNS is a distributed database that maps familiar Internet domain names, such as `mirage.zoo.net`, to Internet Protocol (IP) addresses, such as 204.101.218.97 (see [RFC1033], [RFC1034] and [RFC1035]). Recently, [RFC2065] has proposed extensions to the DNS that allow for data authentication through digital signatures. The extensions provide for "the storage of authenticated public keys in the DNS," supporting general public key distribution as well as DNS security. In this chapter we briefly review the DNS (for an in-depth look, see [DNS]) and its security extensions and evaluate the DNS PKI in terms of the basic PKI characteristics.

Overview of the DNS

The DNS operates through a hierarchy of domains (see Figure 1818). A *domain* is a collection of *hosts* (machines) that are related in some way. For example, computers at U.S. universities are assigned to the `edu` domain. A hypothetical school named Wazzle University might be assigned the domain `wazzle.edu`, and it could in turn assign separate *subdomains* for each of its departments, such as `falls.wazzle.edu` for their Department of Falling.

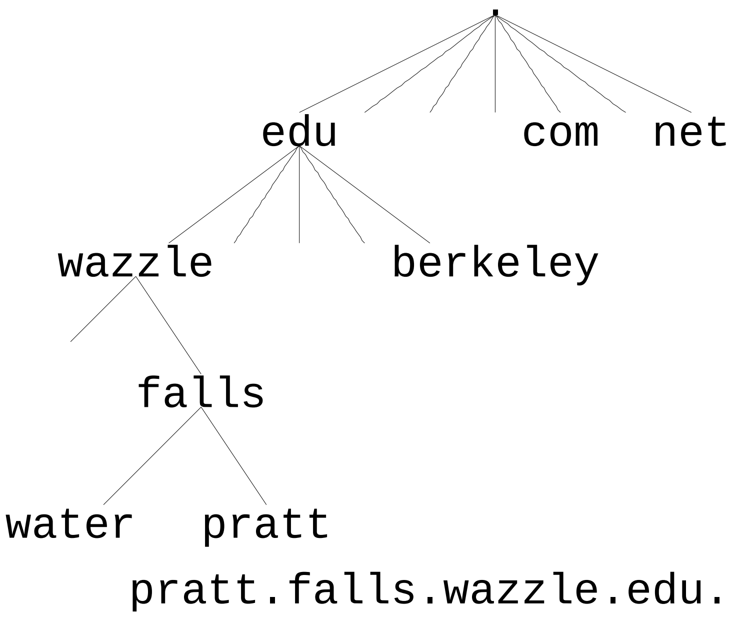

Figure 18 – The DNS hierarchy

Individual hosts in each department would have their respective domain name added to the end of their host name to form a *fully qualified domain name* (FQDN). So the machine named `pratt` in the Falling department would have an FQDN of `pratt. falls.wazzle.edu`.

The root of the DNS hierarchy is aptly called the *root domain* and is denoted by a single dot. To indicate that a host name is an FQDN (rather than being relative to some domain), it is written with a trailing dot to indicate that it's last component is the root domain.

Each domain has one or more *name servers* that hold authoritative information for all the hosts in their domain, including the name servers of subordinate domains. Since these sub-domain name servers are in fact authoritative for the hosts in their domains, the DNS uses the term *zone* to distinguish between a domain and the set of hosts that a name server is fully authoritative for. For example, the *domain* `wazzle.edu` comprises all the hosts at Wazzle University, while the *zone* `wazzle.edu` consists only of those hosts whose names are directly held on the `wazzle.edu` name servers. The Falling Department has its own zone for host names residing on the name servers for its `falls.wazzle.edu` domain.

When an application needs to find an IP address for a given host name, say `pratt.falls.wazzle.edu`, it contacts its local name server which searches the DNS database on the application's behalf (this process is called *resolving*). The search starts by querying one of the root domain's name servers for the address of `pratt. falls.wazzle.edu`. The root name server knows that it is not authoritative for that name, but it does know the addresses of the name servers for the `edu` domain, so it returns those. The query then proceeds to one of these `edu` domain name servers, which will return information about the `wazzle.edu` name servers. Those in turn will provide the addresses of the `falls.wazzle.edu` servers, which will at last provide the address of `pratt.falls.wazzle.edu`.

To make the queries more efficient, the application's name server will save the address information it discovers in its local cache. That way, if the application makes a query about a different host in the `wazzle.edu` domain, its server can contact the `wazzle.edu` name server directly without going through the whole process again. The server does not keep cached information forever. Rather, each datum is assigned a *time-to-live* (TTL) by it's source, and the cache entry is kept until the TTL runs out.

While the primary function of the DNS is to provide a mapping between host names and IP-addresses, the database is flexible enough to provide a wide variety of information. Each entry in the DNS is called a *resource record*, or RR. An RR consists of an *owner name* (the DNS name associated with the RR), a class, a type, and some type-dependent data. A DNS name is always a dot-delimited series of strings. It can represent a zone, a host, a user,[17] or some other entity

[17] In which case the first dot of the name can be thought of as the @ symbol for email addresses. Thus the user marc@aardvark.zoo.net would have the DNS name marc.aardvark.zoo.net.

such as a telephone number.[18] The most common class by far is the IN, or Internet, class. There can be several types of RR in a given class. Some RR types in the IN class include type A for IP-address information, type NS for name server information, and types KEY and SIG which form part of the DNS security extensions that we will now describe.

DNS Security Extensions

[RFC2065] proposes extensions to the DNS that provide key distribution (using the KEY RR) and data origin authentication (using the SIG RR).

The KEY resource record allows public keys to be associated with DNS names. Because of the relatively unsophisticated DNS name format, the KEY RR contains flag bits that indicate the kind of DNS name the RR's owner name represents. The flag bits also indicate possible usage restrictions on the key ("do not use for authentication" or "do not use for confidentiality"), whether there is a key or not (for example, a zone with a "no-key" KEY RR indicates that the zone is not secured), and whether the key can be used with the IPSEC and/or MIME email security protocols.[19] The KEY RR also contains a Protocol field to indicate whether a key can be used for a protocol not covered in the flag bits.

The SIG resource record is designed to "unforgably authenticate other RRs [including KEY RRs] of a particular type, class, and name" binding them to a time interval and the signer's domain name. The SIG RR, together with the RRs it signs, is in effect the secure DNS certificate.

[RFC2065] expects that in most cases a single private key (the "zone key") will generate all the SIG RRs for a given zone. The zone would act as its own CA, its authority coming from its super-zone. In a typical zone configuration, there would be a number of A, NS, KEY and other types of resource records. All the RRs with the same name, class and type would be signed by a single SIG RR.

---

18 See [RFC1530].

19 See [RFC1825] and [RFC1847].

The SIG RR's owner name indicates what name is covered by the SIG. The SIG RR's type-specific data contains, in addition to the actual signature and some other information, the type covered by the signature, when the signature was made and when it will expire, and the signer's DNS name. The signer's name is usually the zone that contains the RRs being signed.

Resolving authenticated data in the DNS involves starting with one or more trusted public keys obtained via some out-of-band means. Given the (trusted) public key for a zone, it is possible to obtain the keys for its sub-zones as well as the key for its super-zone. This makes it possible to securely travel the domain hierarchy without needing to start with the root key.

Secure DNS in Action

Although the primary goal of the secure DNS is to provide a secure mapping between DNS names and IP addresses, it can be extended to provide a general public key distribution service for other protocols. Extensions are already defined for email and secure-IP, and others can easily be defined. What follows is an example of how the secure DNS can be used for encrypted and authenticated email.

Alice (`alice@foo.com`) wishes to send a signed, encrypted message to Bob (`bob@bar.edu`). She requires Bob's public key to encrypt the message, so she contacts her local DNS server (arrow 1 in Figure 1919) with a request for all the KEY re-

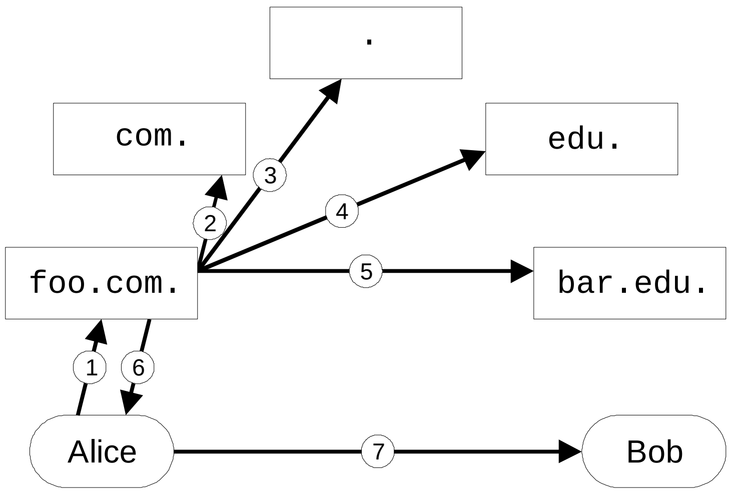

Figure 19 – Secure DNS example

source records of `bob.bar.edu`. The `foo.com` server resolves the query almost as it would any other. Normally, it would contact the root domain server. However, in order to maintain security, it must know the root's key prior to making any requests to the root server. It can either be pre-configured with the root key or not, as we have assumed here. The secure DNS mandates that every secure zone sign the key of its super-zone. This allows the tree of domains to be climbed, bypassing the direct need for the root key.

Since the `foo.com` server does not know the root key, it contacts the `com` domain server requesting the KEY record for `com`'s super-zone (arrow 2). Once the root key is obtained, the query proceeds in a similar fashion to a regular DNS query, except that KEY and SIG records are also returned with the IP addresses of sub-domain servers. That is, when the root server is queried for the KEY record of `bob.bar.edu` (arrow 3), it returns the IP address records for the server of the `edu` domain as well as the KEY record for that domain. Also, associated with the IP address records is a SIG record that digitally signs the address records. Any KEY records also have a related SIG record. For strong authentication to be maintained, the server must return these SIG records along with the IP and KEY records, and the resolving server (`foo.com` in this case) should verify the signatures.

Thus when the `edu` domain server is queried (arrow 4) it returns the IP address for the `bar.edu` server as well as the key for the `bar.edu` zone and their associated SIG records. The query to the `bar.edu` server (arrow 5) only returns a (signed) set of KEY records, since there are no IP addresses associated with `bob.bar.edu`. The `foo.com` server then answers Alice's query with the securely-obtained set of keys for `bob@bar.edu` (arrow 6). Bob's email key is determined by the appropriate flag in the KEY record, so Alice can select the right key from the set and use it to encrypt her email for Bob (arrow 7). Bob can verify Alice's signature on the message by obtaining her signature-verification public key in the same way.

The secure DNS allows for cross-certification between zones, eliminating the need for a root key. For example, the `foo.com` and `bar.edu` servers could have cross-certified each other, eliminating the need for queries 2, 3 and 4 in the protocol described above. Such cross-certification is impractical on a large scale, but it does allow for an organization with several domains to set up a secure DNS in the absence of keys for the higher-level domains.

The Secure DNS PKI

We now describe the secure DNS PKI in terms of the basic PKI characteristics.

The Secure DNS PKI Characteristics	
Certificate information	DNS certificates can contain any kind of resource record.
CA arrangement	Each DNS CA corresponds to a DNS zone, so the CAs are arranged according to the domain name hierarchy. Any CA can certify the key of any other CA. This forms a general hierarchy with cross-certificates.
CA « Subject « User relationship	The subject of a DNS certificate can be any entity that can be assigned a DNS name. Thus the CA of a subject is determined by which Internet domain the subject exists under. When the subject does not have her own, distinct domain, she must have a close relationship with her CA. When the subject does have her own domain, she is her own CA.

CA « Subject « User trust relationships	Each user is expected to fully trust at least one CA. The DNS's hierarchical CA structure lets one know if an entity's certification is derived from a particular CA (via the entity's DNS name). However, there is little a CA can do once it has certified a sub-zone – the sub-zone has full authority over its domain and the parent CA has no means of constraining its trust. For example, a sub-zone can cross-certify an untrustworthy domain that its parent would not, yet users in the sub-zone might believe that they are still following the policies of its parent.
Certificate validation method	DNS certificates are validated online. Their validity period is defined by both the TTL and certificate's expiration date. If either indicates an expired certificate then the certificate must be revalidated. The TTL mechanism is used to reduce the online bandwidth requirements of the DNS, however it does present a window of opportunity for a revoked certificate to be falsely considered as valid.
Certificate revocation method	Certificates are revoked when an entity indicates to its CA that some of its information has changed (e.g. a user has changed their email key) and the CA updates the appropriate entries in its server. The time it takes for the change to propagate throughout the system is at most the largest TTL of the changed entries.
Identity vs. credential certificates	The DNS certificate identifies the owner of a public key by assigning the key a DNS name. Credential certificates are not defined, although credential-serving systems can be built atop the secure DNS.
Irrefutability and strong authentication	Strong authentication is provided as long as a DNS name resolution only passes through secure zones. The more such zones a resolution passes through, the less its result can be trusted, so the secure DNS requires that if a shorter resolution path is found, then any longer ones should be discarded in its favor.

In-band vs. out-of-band authentica-tion	Each resolving application needs to be initialized with at least one trusted key obtained via some out-of-band means. This key is usually the zone key of the domain in which the resolver resides.
Anonymity	Anonymity was not a design goal of the secure DNS, although it does not make anonymity impossible to achieve.

Table 5 – Basic Characteristics of the DNS PKI

Conclusions

The secure DNS extensions are very new and time will tell whether they succeed in providing a general PKI that does more than add security to the DNS database. There are several issues that need to be addressed before the secure DNS can be adopted as a ubiquitous PKI. Among them are:

- The structure of the domain name hierarchy. Already the Internet is encountering limitations in the naming system. New level-one domains (such as `.edu` and `.com`) are about to be adopted to provide more flexibility and relieve the demand on some servers. However, the system was not designed to be used as part of a generic key distribution service. One can easily imagine entities that exist under several different domains. How should they manage their keys in the secure DNS?
- Reliance on the root key. Cross-certification can only go so far. Eventually, a ubiquitous secure DNS will require a root key. This would necessitate that a great deal of trust be placed in that key, making it a tempting target for security compromise. If the secure DNS were to be adopted for everything from email to electronic commerce, it is doubtful that the root key could be made secure enough.
- The current secure DNS is essentially an identity-based certification system. Although it is possible to build a credential-serving system around it, there is currently no effort to do so. For that matter, the extensibility of the

system is limited. For instance, the specification allows for a key to be designated for up to 254 different protocols. Is that enough?

- There is no way to constrain trust. When a sub-zone is certified, it is free to certify anything it wishes, and its parent is powerless to prevent abuses. The assumption seems to be that as one travels down the domain hierarchy, one gets closer to (i.e. interacts more with) the actual users and so the users would tend to trust the lower domains that are closest to them more than the higher domains. However, there are many instances where this is not the case. For example when a corporations certifies parts of itself (such as its own divisions or departments) it wants to make sure that those parts do not stray from their assigned roles.
- The time-to-live mechanism. TTLs are required in the regular DNS because queries are very frequent and caching is necessary to avoid overwhelming the system. However, it is questionable whether the TTL paradigm can be made to work well in a PKI, especially as the system is called upon to authenticate more sensitive data.
- Finally, there are many Internet users who do not have a close relationship with their DNS server. Most clients of Internet Service Providers (ISPs) generally do not maintain their own DNS server, and so must rely upon their ISP to keep their DNS entries current. This reliance will become crucial if the DNS starts to serve as a general PKI, and many users might not be comfortable with such a relationship.

These issues, while unresolved, do not prevent the secure DNS from being used as a practical PKI in many situations. Nor do these problems present an impenetrable barrier to widespread adoption of the secure DNS as a generic PKI. As we noted above, time will tell.

Chapter 8

CREDENTIAL-BASED PKI SYSTEMS

Much recent work has focused on moving away from identity-based PKIs to a more general system based on attributes or credentials. At present, there are three main proposals for this kind of system: ANSI draft X9.45, the Simple Distributed Security Infrastructure (SDSI), and the draft Simple Public Key Infrastructure (SPKI). This chapter focuses mainly on SDSI. We touch briefly on the SPKI effort, but as it is still quite new there is little that can said of it as yet. We do not cover the X9.45 work, as we were unable to obtain any of its documentation and so are forced to merely acknowledge its existence here.

Simple Distributed Security Infrastructure (SDSI)

SDSI (pronounced “sudsy”) was created by Ron Rivest and Butler Lampson and is described in [SDSI]. SDSI is designed to “facilitate the construction of secure systems” and “provides simple, clear terminology for defining access-control lists and security policies.” It is also an attempt to move away from identity-based certification and towards a system based on roles and credentials. There are, as yet, no actual systems based on SDSI.

The SDSI system is “key-centric.” Rather than attach a public key to an identity, SDSI entities are the keys themselves. Specifically, SDSI calls it’s entities “principals” and defines them to be digital signature verification keys. The idea is that the key is a “proxy” for the individual who controls it’s associated private key. Thus SDSI principals are public keys that can make declarations by issuing verifiable signed statements.

SDSI Certificates

Those signed statements come mainly in the form of certificates. SDSI provides for three types of certificates, and any principal can create any kind of certificate. In no particular order, the three certificate types are:

- Identity certificates
- Group-membership certificates
- Name-binding certificates

SDSI identity certificates bind some identifying information to a principal. The main goal of a SDSI identity certificate is to allow a human reader to identify the individual behind a principal. As such, the certificates are designed to be human-friendly, usually containing some free-form text and perhaps a photograph or other information. Machine-readable tags, such as OIDs, are not used as SDSI's designers believe that determining the identity behind a principal will almost always involve a human anyway.

Identity certificates play a relatively small role in the SDSI system. More important are group-membership certificates which assert that a principal does or does not belong to some group (more on SDSI groups below), and name-binding certificates which bind a name to some value (typically, but not necessarily, a principal).

SDSI Names

When a principal creates a certificate binding a name to some value, that name is said to exist in the principal's *local name space.* Each principal can create his own local names which he can use to refer to other principals. The names are arbitrarily chosen – there is no naming system to follow, and no attempt is made make names that are "globally" unique across all local names spaces. Thus some principal which Alice has named bob may be completely different from the principal that Carl calls bob.

SDSI provides a simple method to *link* local name spaces together. If Alice has named a principal `bob`, and Bob has named another principal `jim`, then Alice can refer to that second principal as `bob’s jim`. Alice can refer to any of `bob`’s principals in this way, and the chain can be extended indefinitely, for example as in `bob’s jim’s mother’s doctor`. Names can also be “symbolically” defined. For example, Alice’s local name `bob` can denote `company’s Bob-Smith`. If the principal that Alice calls `company` changes the principal it calls `Bob-Smith`, then the principal that she calls `bob` changes as well.

SDSI achieves this name linking because it has an “on-line” orientation. Principals that issue certificates are assumed to be able to provide an on-line Internet server to distribute those certificates upon request. Thus for Alice to find the actual principal behind the name `bob’s jim`, she simply connects to `bob`’s server and requests the name-binding certificate that defines the name `jim`.

SDSI also provides for multiple global name spaces. These are the name spaces defined by a small set of “distinguished root” principals. These principals have special reserved names (that end with `!!`) which are bound to the *same* principal in *every* name space. SDSI does not describe how this is achieved in any detail. However, it does give SDSI the power to access “standard” name spaces, for example `VeriSign!!’s Microsoft’s CEO` or `DNS!!’s com’s microsoft’s “Bill Gates”`.[20] Here, the name `VeriSign!!` evaluates to the *same* principal in *all* name spaces. The name `DNS!!` also resolves to another, unique principal in all name spaces. Note that this does not mean that all principals have a single, unique global name. Rather, a principal can have multiple global names that start from different distinguished roots (as in our example).

SDSI Groups

SDSI allows its principals to define groups, or sets, of principals. Each group has a name and a set of members. The name is local to some principal, which is

[20] The names `VeriSign!!` and `DNS!!` are taken from the SDSI paper ([SDSI]). `DNS!!` represents names defined under Secure DNS (the `DNS!!` principal represents the DNS root key), while `VeriSign!!` denotes names created by VeriSign Inc., a U.S. certification authority.

the “owner” of the group. Only a group’s owner may change its definition. A group can be an explicit list of the group’s members (either as a list of principals and/or names of principals), or it can be defined in terms of other groups. Any principal can define his own groups and export them via his servers in much the same way as name bindings. The servers can issue membership certificates based on the groups’ definitions.

Groups provide the fundamental mechanism by which SDSI operates. When defining a security policy (for example, specifying who is allowed access to a particular resource), SDSI allows you to define the group of authorized principals, then place the group’s name on the resource’s access-control list(s). SDSI’s naming system allows a person to easily understand security policies created in this way.

SDSI in Action

To better illustrate SDSI’s ideas, we now provide a small example of how SDSI would operate in a typical situation. SDSI defines *protocols* in which *messages* are exchanged. Our example, illustrated in Figure 2020, shows how the SDSI `Membership` and `Get` protocols are used to access an FTP server.

The FTP server is administered by Jim, an employee of ABC Inc. Jim wants to give FTP access to his friends and to other ABC employees. Jim defines a group called `ftp-users` on his SDSI server. That group contains two entries, the groups named `friends` and `abc’s employees`, meaning that for a principal to be a member of the `ftp-users` group it must either be a member of `friends` or a member of `abc’s employees` (or both). Jim has also defined a group he calls `friends` on his server, which contains the names `alice`, `stanley` and `laurel`, corresponding to the principals of Jim’s friends. Furthermore, Jim has bound the name `abc` to ABC Inc.’s principal. Finally, ABC Inc. has created a group it calls `employees` on its SDSI server, which lists all the principals of its employees, including one that they have named `BobSmith103456`, which is Bob’s principal. These group definitions are shown in Figure 2121.

We begin our example by illustrating how Jim's friend Alice gains FTP access, then follow with the more complicated example of how Bob gains the same access. The messages sent and received by Alice are depicted in Figure 2020 with white-headed arrows, while those involving Bob are shown with black-headed arrows.

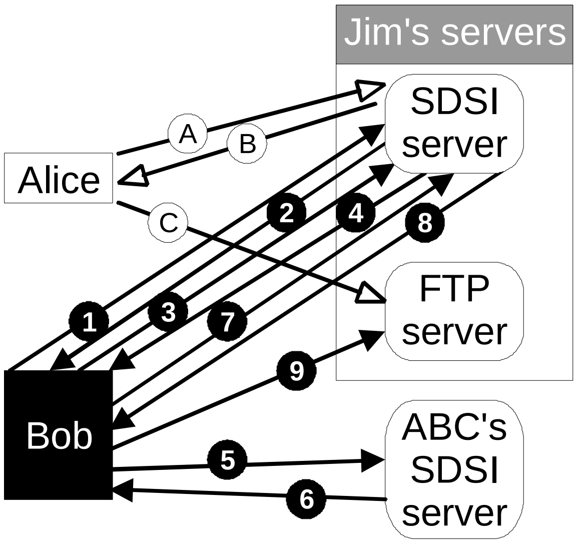

Figure 20 – SDSI protocol example

To gain access to the FTP server, Alice must show that she is a member of Jim's `ftp-users` group.[21] She sends a SDSI `Membership.Query` message (arrow A in Figure 2020) to Jim's SDSI server, in which she specifies her principal and the group name `ftp-users`. The message is a request for a certificate stating the membership status of the given principal for the given group. That status may be one of `true` (i.e. the principal is a member), `false` (is not a member) or `fail` (may or may not be a member, additional credentials are needed for a full determination).

In Alice's case when Jim's SDSI server performs the membership check it

Jim's Groups	
`ftp-users`	`friends`

`friends`
`abc's employees`
	`alice`
	`stanley`
	`laurel`

ABC's Groups
`employees`

…
`BobSmith103456`
`JimJones157638`
…

Figure 21 – Sample SDSI groups

[21] She knows this from prior contact with the FTP server. This is not illustrated.

finds that the principal that Jim has named `alice` matches the principal in the `Membership.Query` message and is a member of Jim’s `friends` group, which satisfies the membership requirements for the `ftp-users` group. Jim’s SDSI server replies to Alice’s query with a `true` membership certificate for Alice’s principal (arrow B). Alice then presents the membership certificate to Jim’s FTP server (arrow C) to gain access.

Bob’s case is a bit more complicated. Bob is an employee of ABC Inc. but his principal is not a member of Jim’s `friends` group. When Bob sends a `Membership.Query` to Jim’s SDSI server (arrow 1), the reply (arrow 2) is a `fail` membership certificate along with an indication that if Bob can show membership (or non-membership) in Jim’s `abc’s employees` group it would help in determining his membership in the `ftp-users` group.

Bob needs to find out which principal Jim has named `abc`,[22] so he sends a SDSI `Get` protocol `Get.Query` message to Jim’s SDSI server (arrow 3). The `Get` protocol is used to retrieve certificates from a server. In this case, Bob requests all of Jim’s name-binding certificates that specify the local name `abc`. Jim’s SDSI server replies with a certificate showing that Jim’s local name `abc` corresponds to ABC Inc.’s principal (arrow 4).

Bob now contacts ABC’s SDSI server with a `Membership.Query` message for the `employees` group (arrow 5). ABC’s SDSI server finds that Bob’s principal is a member of the group, and returns a `true` membership certificate (arrow 6). Now Bob can send another `ftp-users` `Membership.Query` message (arrow 7) to Jim’s SDSI server, this time including the membership certificate he obtained form ABC’s SDSI server. Using this new credential, Jim’s SDSI server can verify that Bob is a member of the `ftp-users` group and return a `true` membership certificate (arrow 8) which Bob can present to the FTP server to gain access (arrow 9).

[22] It is not necessarily ABC’s principal, as Jim is free to assign any name he chooses to any principal. In our example Jim did name ABC’s principal `abc`, but he could have named it `xyz` or `big-green` or any other string.

Key Management Under SDSI

SDSI provides a simple and elegant system for publishing, modifying and withdrawing signed statements such as group membership certificates. However, it fails to present a coherent method to manage the public keys on which it is based. We now examine the implications of key revocation under SDSI.

For example, let us define our own `ftp-users` group as a set of three principals we have named `alice`, `bob` and `carl`. How we define those names is important when it comes to handling the revocation of one of the principals. If we have our own copy of each principal on our server then our names will resolve directly to a local copy of a principal. This arrangement would force us to keep track of `alice`, `bob` and `carl`, in case one of them changes their key for some reason, so we can keep our local copies up to date.

To save ourselves some work, we could define two of the names in terms of the third's name space. So our definition for `bob` could be `alice`'s `bob`, and `carl` could be `alice`'s `carl`. Now we only have to keep a local copy of `alice`'s principal on our server. Also, we only need to worry about `alice` changing her key, as long as she faithfully keeps track of `bob` and `carl` for us.

In a sense, `alice` is acting as our CA, but in fact this is very similar to PGP's web of trust model, and suffers from similar drawbacks. What if `alice` doesn't bother to keep track of `bob` and `carl`? Or perhaps `alice` has also delegated her name for `bob` in the same way we have, which would force us to rely on someone of `alice`'s choosing for our name space integrity. Or, worse, what if `alice` decides to call some other principal `bob` without telling us? We can't really trust `alice` unless she agrees to be our CA in some official capacity.

Clearly, SDSI's linked local name spaces do not provide for adequate key management. Global name spaces can solve this problem, as the entity that serves as a global root can assume the ultimate responsibility for ensuring that its names are accurate. However, this would require one or more fixed points of global trust, which has proven to be infeasible. Also, a satisfactory global naming scheme has yet to be defined.

SDSI's global distinguished roots are designed to provide "standard" name spaces, but only so far as to provide globally recognizable synonyms for principals. SDSI specifically does not address liability issues with respect to who is responsible for ensuring the integrity of a global name space, nor does SDSI provide any mechanisms for doing so.

The SDSI PKI

Table 66 describes the SDSI PKI in terms of the basic PKI characteristics.

SDSI PKI Characteristics	
Certificate information	Identity certificates are free-form and extensible. SDSI's group mechanism allows any kind of attribute to be attached to a principal.
CA arrangement	Any SDSI principal can be a CA, and no principal is subordinate to another in terms of what kind of certificates it can issue. There is no assigned arrangement for SDSI CAs.
CA « Subject « User relationship	SDSI makes no distinction between CAs, certificate subjects or certificate users. All are principals with equal authority and powers.
CA « Subject « User trust relationships	SDSI trust relationships are very similar to PGP trust relationships. Each SDSI principal decides how much she can trust any other principal. Although SDSI does not provide an automatic way for a principal to prevent her trust from being infinitely extended, it does make trust relationships explicit and auditable, so a principal can tell if her trust has been misplaced.
Certificate validation method	SDSI operates in a completely online environment. Certificates can be revalidated with each use, or they can be created with a "reconfirmation period" such as "every two weeks" or "once an hour".

Certificate revocation method	SDSI's online orientation allows principals to revoke certificates instantaneously, and the reconfirmation periods place a clear upper bound on how long revocation information will take to propagate.
Identity vs. credential certificates	SDSI provides both identity and, through its group mechanism, credential certificates.
Irrefutability and strong authentication	SDSI's linked local name spaces can dilute the strength of authentication as name space chains grow. Longer chains require a greater number of trustworthy principals. SDSI's global name space mechanism allows for strong authentication provided there are a small number of distinguished global roots.
In-band vs. out-of-band authentication	Users must obtain the principals for the global distinguished roots via some out-of-band means. Once that is achieved, all other information can be obtained in-band.
Anonymity	SDSI can provide anonymity for its principals at the expense of authentication strength. To support strong authentication, a principal must either provide a global name for itself, or be well-known to the other principals he deals with.

Table 6 – Basic Characteristics of the SDSI PKI

The Simple Public Key Infrastructure

At the beginning of 1996, just before the publication of the SDSI paper, an Internet working group was formed to propose an alternative PKI to the X.509v3-based PKIX. This new group is called the Simple Public Key Infrastructure (SPKI) Working Group. So far, the group has only published a requirements statement, [SPKI], and a draft certificate format, [SPKC]. Much of what follows is taken from [SPKI].

There are several similarities between the SPKI and SDSI. In particular, one of the SPKI's requirements is to support, where possible, the SDSI local name space mechanism. SDSI is, and the SPKI will be, key-centric (SDSI speaks of "principals" while the SPKI uses the term "keyholders"), and both provide a mechanism for attaching credentials (the SPKI calls them attributes) to public key values (SDSI through its groups, the SPKI by issuing certificates).

Although the SPKI will use SDSI names, it considers global naming schemes to be irrelevant. To quote the SPKI requirements document: "A user of a certificate needs to know whether a given keyholder has been granted some attribute and that attribute rarely involves a name." The SPKI recognizes the need to uniquely identify keyholders, and considers the public key value itself (or its hash) adequate for that purpose.

The SPKI will be a credential-based system. Its certificates will carry the minimum attributes necessary to get a job done. This is to protect, as much as possible, the privacy of keyholders. Using monolithic certificates that contain many attributes, most of which are irrelevant in a given situation, would reveal more information about the keyholder than he might like. Also, to discourage keyholders from sharing their private key values, the SPKI will allow a certificate holder to delegate the attributes she acquires from the certificate. Finally, SPKI certificates are to have several validation and revocation mechanisms: validity periods, periodic reconfirmation, CRLs, or some user-defined conditions to be tested online or through other certificates.

Conclusions

SDSI and the SPKI are ambitious efforts to create a credential-based certification system. While there is clearly a need for such a system on the Internet, it is too early to tell if a particular paradigm will succeed. Some kind of naming system is required if people are to make sense of the system, but the viability of SDSI's linked local name spaces, and the need for global names, remain open questions.

About all that is certain at this point is that identity-only systems such as X.509 and PGP are inadequate as general-purpose PKIs.

Chapter 9

CONCLUSION

This thesis has presented a unified, coherent method for comparing and evaluating diverse PKI systems and has applied it to a few real-world examples of such systems. The basic PKI characteristics described here can help PKI designers define their goals and recognize where tradeoffs have been made. The framework can also aid PKI implementers in choosing which PKI system best suits their needs. We hope it proves useful to all involved in the PKI field.

The framework is intended to cover the major aspects of all PKI systems. As such, it will most likely require periodic revisions to keep it in step with the latest PKI developments. In serving as a guide, the basic characteristics are meant to enhance rather than restrict new PKI work.

GLOSSARY

ANSI	American National Standards Institute
CA	Certification Authority
CEO	Chief Executive Officer
CN	Common Name
CRL	Certificate Revocation List
DES	Data Encryption Standard
DIT	(An X.500) Directory Information Tree
DN	(An X.500) Distinguished Name
DNS	Domain Name System
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
IDEA	International Data Encryption Algorithm
IP	Internet Protocol
IPRA	Internet Policy Registration Authority
IPSEC	Internet Protocol SECurity extensions
ISO	International Organization for Standardization
ISP	Internet Service Provider
ITU	International Telecommunication Union
LOTLA	List of TLAs
MD5	Message Digest 5

MIME	Multipurpose Internet Messaging Extensions (also known as Multimedia Internet Mail Extensions)
MIT	Massachusetts Institute of Technology
OID	Object Identifier
PCA	Policy Certification Authority
PEM	Privacy Enhanced Mail
PGP	Pretty Good Privacy
PKI	Public-Key Infrastructure
PKIX	Public-Key Infrastructure using X.509
RDN	(An X.500) Relative Distinguished Name
RFC	Request For Comments
RR	Resource Record
RSA	Rivest, Shamir and Adleman
SDSI	Simple Distributed Security Infrastructure
SET	Secure Electronic Transaction
SHA	Secure Hash Algorithm
SPKI	Simple Public Key Infrastructure
TLA	Three-Letter Acronym
TTL	Time-To-Live
WWW	World-Wide Web

BIBLIOGRAPHY

Note: An online version of this bibliography is maintained by the author at `http://www.zoo.net/~marcnarc/PKI/References.htm`. This web page maintains links to online copies of the actual documents, where possible.

[BFL96] Blaze, M., Feigenbaum, J. and Lacy, J., "Decentralized Trust Management." *Proceedings of the IEEE Conference on Security and Privacy.* May 1996.

[DES] *Data Encryption Standard.* Federal Information Processing Standards publication 46-2. December 1993.

[DH76] Diffie, W. and Hellman, M. E., "New directions in cryptography." *IEEE Transactions on Information Theory,* 22(1976), pp. 644-654.

[DNS] Liu, C. and Albitz, P. *DNS and BIND.* O'Reilly & Associates, Inc. 1992.

[El85] ElGamal, T. "A public key cryptosystem and a signature scheme based on discrete logarithms." *IEEE Transactions on Information Theory,* 31 (1985), pp. 469-472.

[Elli] Ellison, C. M., *Generalized Certificates.* September 1996.

[Fa96] Fancher, C. H., "Smart Cards." *Scientific American*, vol. 275, no. 2 (August, 1996), pp. 40-45.

[Fo95a] Ford, W., "Advances in Public-Key Certificate Standards." *ACM SIGSAC Security Audit & Control Review*, vol. 13, no. 3. July 1995.

[Fo95b] Ford, W., *A Public Key Infrastructure for U.S. Government Unclassified but Sensitive Applications.* Produced by Nortel and BNR for NIST. September 1995.

[FoBa] Ford, W. and Baum, M. *Secure Electronic Commerce: Building the Infrastructure for Digital Signatures and Encryption.* Prentice Hall, 1997.

[Fr96] Froomkin, M., *The Essential Role of Trusted Third Parties in Electronic Commerce.*

[IHAC] *The Challenge of the Information Highway: Final Report of the Information Highway Advisory Council.* Government of Canada. 1995.

[McBu] McBurnett, N. *PGP Web of Trust Statistics.*

[Ne95] Negroponte, N., *Being Digital*. Alfred A. Knopf, Inc. 1995.

[NIST1] *Public Key Infrastructure Study Final Report.* Produced by the MITRE Corporation for NIST. April 1994.

[NIST2] Polk, W. (Ed.), *Federal Public Key Infrastructure (PKI) Techincal Specifications (Version 1) Part A: Requirements.* NIST PKI Technical Working Group. January 1996.

[NIST3] Nazareno, N. (Ed.), *Federal Public Key Infrastructure (PKI) Technical Specifications (Version 1) Part B: Technical Security Policy.* NIST PKI Technical Working Group. January 1996.

[NIST4] Burr, W. (Ed.), *Federal Public Key Infrastructure (PKI) Technical Specifications (Version 1) Part C: Concept of Operations.* NIST PKI Technical Working Group. November 1995.

[NIST5] *Federal Public Key Infrastructure (PKI) Technical Specifications (Version 1) Part D: Interoperability Profiles.* Produced by CygnaCom Solutions, Inc. for the NIST PKI Technical Working Group. September 1995.

[NIST6] Trcek, D. and Blazic, B. J., *Certification Infrastructure Reference Procedures.* NIST PKI Technical Working Group (W. Burr, Ed.), NISTIR 5788, NIST. September 1995.

[NIST7] Baum, M. S., *Certification Authority Liability and Policy.* NIST-GCR-94-654, NTIS Doc. No. PB94-191-202 (Springfield, VA: National Technical Information Service, 1994).

[PKCS0] Kaliski, B. S. Jr., *An Overview of the PKCS Standards.* RSA Laboratories. November 1993.

[PKCS10] *PKCS #10: Certification Request Standard.* RSA Laboratories. November 1993.

[PKCS6] *PKCS #6: Extended-Certificate Syntax Standard.* RSA Laboratories. November 1993.

[PKCS9] *PKCS #9: Selected Attribute Types.* RSA Laboratories. November 1993.

[PKIX1] Housley, R., Ford, W. and Solo, D., *Internet Public Key Infrastructure Part I: X.509 Certificate and CRL Profile.* IETF X.509 PKI (PKIX) Working Group (draft).

[PKIX3] Farrell, S., Adams, C. and Ford, W., *Internet Public Key Infrastructure Part III: Certificate Management Protocols.* IETF X.509 PKI (PKIX) Working Group (draft).

[RFC1032] Stahl, M. *Domain Administrators Guide.* November 1987.

[RFC1033] Lottor, M. *Domain Administrators Operations Guide.* November 1987.

[RFC1034] Mockapteris, P. *Domain Names – Concepts and Facilities.* November 1987.

[RFC1035] Mockapetris, P. *Domain Names – Implementation and Specification.* November 1987.

[RFC1321] Rivest, R. *The MD5 Message-Digest Algorithm.* April 1992.

[RFC1421] Linn, J. *Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures.* February 1993.

[RFC1422] Kent, S. *Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management.* February 1993.

[RFC1423] Balenson, D. *Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers.* February 1993.

[RFC1424] Kaliski, B., *Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services.* February 1993.

[RFC1510] Kohl, J. and Neuman, B. C. *The Kerberos Network Authentication Service (Version 5).* September 1993.

[RFC1530] Malamud, C. and Rose, M. *Principles of Operation for the TPC.INT Subdomain: General Principals and Policy.* October 1993.

[RFC1825] Atkinson, R. *Security Architecture for the Internet Protocol.* August 1995.

[RFC1847] Galvin, J., Murphy, S., Crocker, S. and Freed, N. *Security Multi-parts for MIME: Multipart/Signed and Multipart/Encrypted.* October 1995.

[RFC2065] Eastlake, D. 3rd and Kaufman, C. *Domain Name System Security Extensions*. January 1997.

[RSA78] Rivest, R., Shamir, A. and Adleman, L. “A method for obtaining digital signatures and public key cryptosystems.” *Communications of the ACM,* 21 (1978), pp. 120-126.

[Sa96] Saeki, M., *Elliptic Curve Cryptosystems.* M.Sc. Thesis, McGill University School of Computer Science. 1996.

[Sc96] Schneier, B., *Applied Cryptography, 2nd ed.* John Wiley & Sons, Inc. 1996.

[SDSI] Rivest, R. and Lampson, B. *SDSI - A Simple Distributed Security Infrastructure.* April 1996.

[SET] *Secure Electronic Transaction (SET) Specifications.* MasterCard and Visa. February 1996.

[SHS] *Secure Hash Standard.* Federal Information Processing Standards publication 180-1. April 1995.

[St95] Stinson, D. R., *Cryptography: Theory and Practice.* CRC Press, Inc. 1995.

[SPKC] Ellison, C. M., Frantz, B. and Thomas, B. M., *Simple Public Key Certificate.* July 1996.

[SPKI] Ellison, C. M., *SPKI Requirements.* February 1997.

[TBP96] Trcek, D., Blazic, B. J. and Pavesic, N., "Security Policy Space Definition and Structuring." *Computer Standards & Interfaces*, Vol. 18, No. 2. North-Holland. March 1996. pp. 191-195.

[TKBB94] Trcek, D., Klobucar, T., Blazic, B. J. and Bracun, F., *CA-Browsing System - A Supporting Application for Global Security Services.* ISOC Symposium on Network and Distributed System Security. San Diego, February 1994. pp. 123-128.

[X208] *ITU/ISO Recommendation X.208 – Specification of Abstract Syntax Notation One (ASN.1).* 1988.

[X209] *ITU/ISO Recommendation X.209 – Specification of basic encoding rules for Abstract Syntax Notation One (ASN.1).* 1988.

[X500] *ITU/ISO Recommendation X.500 – Information technology – Open Systems Interconnection – The directory: Overview of concepts, models, and services.* November 1993.

[X509] *ITU/ISO Recommendation X.509 – Information technology – Open Systems Interconnection – The directory: Authentication framework.* November 1993.

[X509a] *ITU/ISO Final text of draft amendments to X.500* | *9594 for certificate extensions.* 1996.

[Zimm] Zimmermann, P. *PGP User's Guide* vol. 1 and 2.